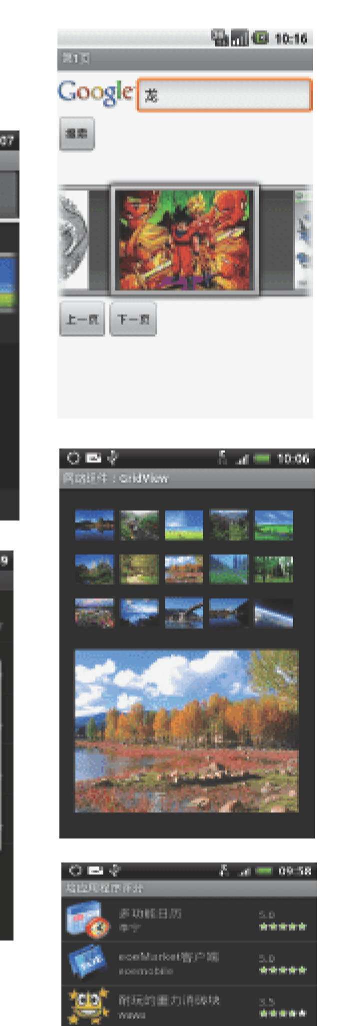
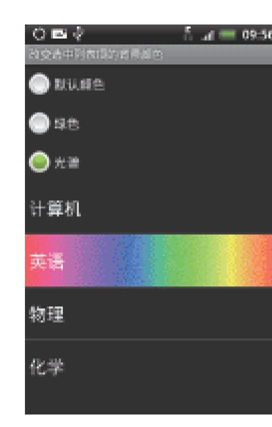
○完全学习手册○



一书在手，快速成为Android开发高手

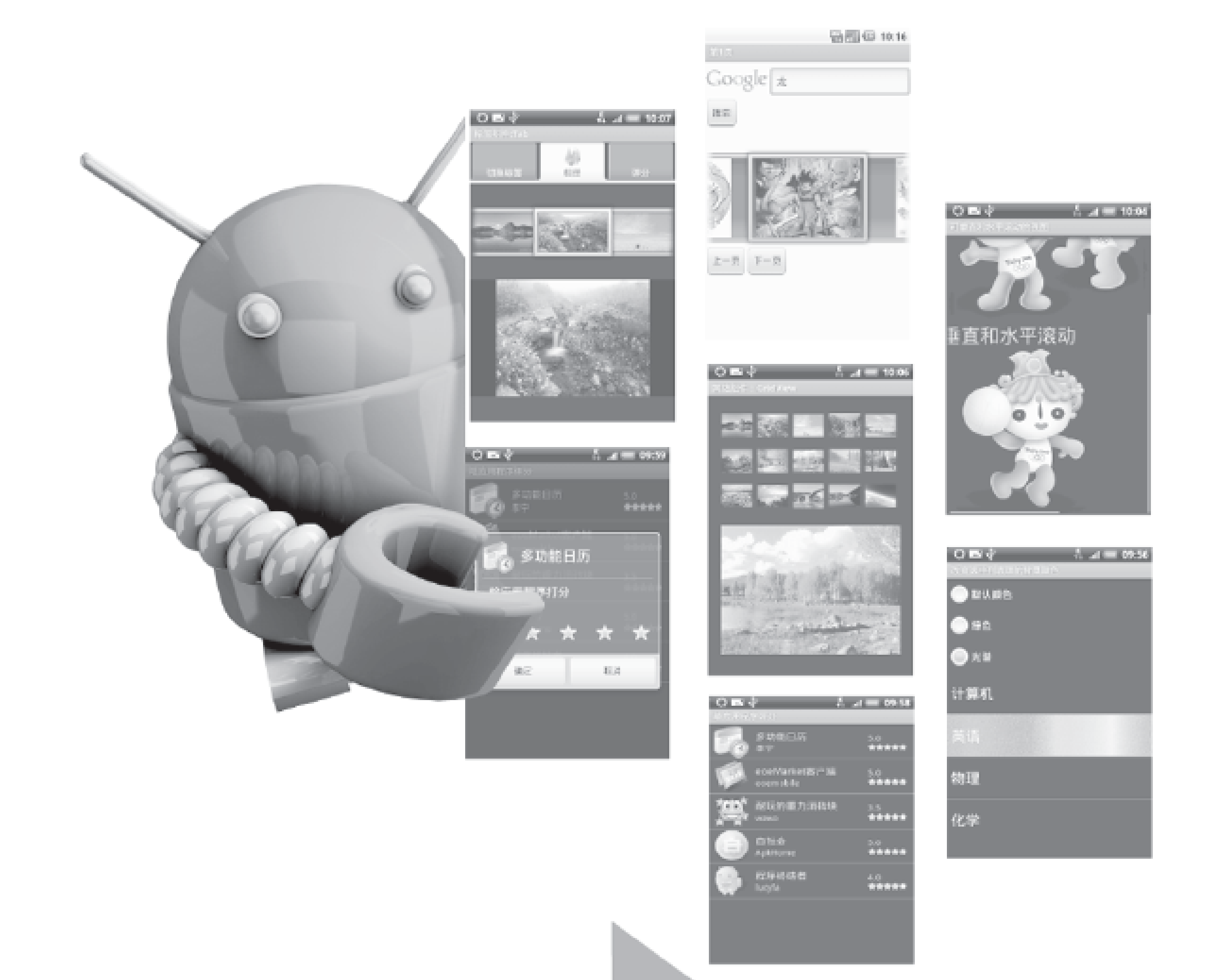
黄永丽　王　晓　孔美云　编著

Android应用开发完全学习手册

基础翔实，**实例丰富**，**图文并茂**，**案例真实**，从基础到实战覆盖Android应用开发的各个领域。

清华大学出版社

○完全学习手册○



Android 应用开发完全学习手册

黄永丽　王　晓　孔美云　编著

清华大学出版社
北　京

内 容 简 介

近年来，Android 的兴起和对移动设备开发领域的冲击已成为热门话题。本书以深入浅出、通俗易懂的方式对 Android 的应用开发进行全面介绍。对于一些较难理解的概念用实例进行说明，这些实例具有较强的针对性，以帮助读者更好地理解各知识点在实际开发中的应用。本书共分为 13 章，内容覆盖了 Android 概述、Android 界面布局、Android 控件、菜单栏与对话框、Intent 和 ContentProvider、Android 的多线程与数据处理机制、Android 数据存储、多媒体应用开发、基于位置服务的应用开发、Android 桌面组件开发，以及传感器的开发等。

本书基础翔实，实例丰富，图文并茂，案例真实，从基础到案例覆盖了 Android 应用开发的各领域，既可作为本科院校、高等职业院校及软件学院计算机类、通信类专业的教材，也可作为相关培训学校的 Android 培训教材及从事 Android 移动编程和应用开发人员的参考用书。

图书在版编目（CIP）数据

Android 应用开发完全学习手册 / 黄永丽，王晓，孔美云编著. —北京：清华大学出版社，2015（2016.8 重印）
（完全学习手册）
ISBN 978-7-302-37617-0

Ⅰ. ①A… Ⅱ. ①黄… ②王… ③孔… Ⅲ. ①移动终端-应用程序-程序设计-手册
Ⅳ. ①TN929.53-62

中国版本图书馆 CIP 数据核字（2014）第 186434 号

责任编辑： 袁金敏
封面设计： 刘新新
责任校对： 徐俊伟
责任印制： 何　芊

出版发行： 清华大学出版社
　　网　　址： http://www.tup.com.cn, http://www.wqbook.com
　　地　　址： 北京清华大学学研大厦 A 座　　**邮　　编：** 100084
　　社 总 机： 010-62770175　　**邮　　购：** 010-62786544
　　投稿与读者服务： 010-62776969，c-service@tup.tsinghua.edu.cn
　　质 量 反 馈： 010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者： 虎彩印艺股份有限公司
经　　销： 全国新华书店
开　　本： 185mm×260mm　　**印　　张：** 18.5　　**字　　数：** 459 千字
版　　次： 2015 年 3 月第 1 版　　**印　　次：** 2016 年 8 月第 2 次印刷
印　　数： 3501～4300
定　　价： 39.00 元

产品编号：054359-01

前　　言

当今社会已经全面进入了移动时代，手机功能越来越智能，越来越开放，为了实现这些需求，必须有一个好的开发平台来支持。2007 年，Google 公司推出了基于 Linux 平台的开源手机操作系统 Android，由于其开放性和优异性，Android 平台得到了业界广泛的支持，是目前最受欢迎的嵌入式操作系统之一，其发展的上升势头势不可挡。

移动终端的快速发展，使得 Android 系统应用的需求激增，很多在校生和广大开发者都加入了 Android 开发阵营。为了帮助开发者更快地进入 Android 开发行列，笔者特意精心编写了本书。本书从读者的实际需求出发，科学安排知识结构，内容由浅入深，循序渐进地逐步展开，具有很强的知识性，反映了当前 Android 技术的发展和应用水平。

全书分 13 章，各章内容介绍如下。

第 1 章介绍 Android 开发基础，内容包括 Android 的发展历史、开发环境的搭建、Android 应用程序组件等。

第 2 章介绍 Android 界面布局及基本控件，内容包括视图 View 概述、线性布局、相对布局、表格布局、文本框及按钮控件等。

第 3 章介绍 Android 控件知识，内容包括 ImageButton 控件、ImageView 控件、单选按钮与复选框、网格视图等。

第 4 章介绍菜单和对话框的使用，内容包括选项菜单和子菜单、上下文菜单、对话框和提示信息等。

第 5 章介绍 Intent 和 ContentProvider 的相关知识，并进行举例说明。

第 6 章介绍 Android 下的多线程与事件处理机制等知识。

第 7 章介绍 2D 应用程序开发，内容包括 SurfaceView、用 2D 技术开发简单游戏、Graphics 类开发及动画实现等。

第 8 章介绍 Android 数据存储的相关知识。

第 9 章介绍多媒体开发，以及使用电话 API 的相关知识。

第 10 章介绍网络与通信，内容包括 HTTP 通信、Socket 网络开发等。

第 11～13 章为综合实例，分别为基于位置服务的应用开发、桌面组件开发及传感器应用开发。

本书通过大量简单易懂的实例使读者快速掌握知识点，每个部分既相互连贯又自成体系，使读者既可以按照本书编排的章节顺序进行学习，也可以根据自己的需求对某一章节进行针对性的学习。同时，本书更加注重知识的实用性和可操作性，通过实例使读者在掌握相关技能的同时学习相应的基础知识。书中所有的实例都已调试运行通过，读者可以直接参照使用。本书知识点全面，结构合理，重点难点突出，实例丰富，语言简洁，图文并茂，适用于 Android 移动软件开发初、中级用户。

本书由黄永丽、王晓、孔美云等老师共同编写，全书由钱慎一、白永刚老师统稿，孔美云老师编写第 2、3 章、黄永丽老师编写了第 4、5 章，张伟伟老师编写了第 6、7 章，王

晓老师编写了第 8、9 章，霍林林老师编写了第 10、11 章，常化文老师编写了第 12、13 章，另外，蒋军军、胡文华、尼朋、聂静、张丽等老师也参与了本书部分内容的编写工作，在此，对他们的辛勤工作表示衷心感谢。最后特别感谢郑州轻工业学院教务处及浙江商业职业技术学院对本书的大力支持。

由于编写时间仓促，加之作者水平有限，书中难免会有错误和疏漏之处，恳请广大读者给予批评指正。

目　　录

第 1 章　Android 开发基础

Android 是基于 Linux 的自由及开放源代码的操作系统，主要使用于移动设备，如智能手机和平板电脑，由谷歌公司和开放手机联盟领导及开发。2012 年 11 月数据显示，Android 占据全球智能手机操作系统市场 76%的份额，中国市场占有率为 90%。本章将介绍 Android 的基本知识及 Android 开发环境的搭建。

1.1　Android 简介

Android 一词的本义指“机器人”，同时也是谷歌公司于 2007 年 11 月 5 日宣布的基于 Linux 平台的开源手机操作系统的名称，该平台由操作系统、中间件、用户界面和应用软件组成。

1.1.1　发展历史

2003 年 10 月，安迪 • 鲁宾等人创建 Android 公司，并组建 Android 团队。

2005 年 8 月 17 日，谷歌低调收购了成立仅 22 个月的高科技企业 Android 及其团队。安迪 • 鲁宾成为谷歌公司工程部副总裁，继续负责 Android 项目。

2007 年 11 月 5 日，谷歌公司正式向外界展示了这款名为 Android 的操作系统，并且当天谷歌宣布建立一个全球性的联盟组织，该组织由 34 家手机制造商、软件开发商、电信运营商及芯片制造商组成，并与 84 家硬件制造商、软件开发商及电信营运商组成开放手持设备联盟（Open Handset Alliance），来共同研发改良 Android 系统，这一联盟将支持谷歌发布的手机操作系统及应用软件，谷歌公司以 Apache 免费开源许可证的授权方式，发布了 Android 的源代码。

2008 年，在 Google I/O 大会上，谷歌提出了 Android HAL 架构图，在同年 8 月 18 号，Android 获得了美国联邦通信委员会（FCC）的批准，在 2008 年 9 月，谷歌正式发布了 Android 1.0 系统，这也是 Android 系统最早的版本。

2009 年 4 月，谷歌正式推出了 Android 1.5 版本，从 Android 1.5 版本开始，谷歌开始将 Android 的版本以甜品的名字命名，Android 1.5 命名为 Cupcake（纸杯蛋糕），该系统与 Android 1.0 相比有了很大的改进。

2009 年 9 月，谷歌发布了 Android 1.6 的正式版，并且推出了搭载 Android 1.6 正式版的手机 HTC Hero（G3），凭借出色的外观设计及全新的 Android 1.6 操作系统，HTC Hero（G3）成为当时全球最受欢迎的手机。Android 1.6 也有一个有趣的甜品名称，被称为 Donut（甜甜圈）。

2010 年 10 月，谷歌宣布 Android 系统达到了第一个里程碑，即电子市场上获得官方

数字认证的 Android 应用数量已经达到了 10 万个，Android 系统的应用增长非常迅速。在 2010 年 12 月，谷歌正式发布了 Android 2.3 操作系统 Gingerbread （姜饼）。

2011 年 1 月，谷歌称每日的 Android 设备新用户数量达到了 30 万部，到 2011 年 7 月，这个数字增长到 55 万部，而 Android 系统设备的用户总数达到了 1.35 亿，Android 系统已经成为智能手机领域占有量最高的系统。

2011 年 8 月 2 日，Android 手机已占据全球智能机市场 48%的份额，并在亚太地区市场占据统治地位，终结了 Symbian（塞班系统）的霸主地位，跃居全球第一。

2011 年 9 月，Android 系统的应用数目已经达到了 48 万，而在智能手机市场，Android 系统的占有率已经达到了 43%。继续排在移动操作系统首位。谷歌将会发布全新的 Android 4.0 操作系统，这款系统被谷歌命名为 Ice Cream Sandwich（冰激凌三明治）。

2012 年 1 月 6 日，谷歌 Android Market 已有 10 万开发者推出超过 40 万活跃的应用，大多数的应用程序都免费。

1.1.2 Android 的特点

Android 作为一个系统，是一个运行在 Linux 2.6 核心上的 Java 基础的操作系统。

Android 应用程序用 Java 开发而且很容易被放置到新的平台上，其他特点包括硬件支持 3-D 加速图形引擎；支持 SQLite 数据库；一个完整的网页浏览器。

如果开发者熟悉 Java 编程或者是任何种类的 OOP 开发，则可以使用用户接口（UI）开发程序。Android 允许使用 UI 开发，而且支持 XML 为基础的 UI 布局。XML UI 布局对普通桌面开发者是一个非常新的概念。

Android 另一个更令人激动和关注的特点是它的样式，第三方应用程序会和系统自带应用程序具有同样的优先权，这是和大多数系统的不同之处，但是给了嵌入式系统程序一个比由第三方开发者创建的线程优先权大的优先执行权。而且，每一个应用程序在虚拟计算机上以一个非常轻量的方式按照自己的线路执行。

除了大量的 SDK 和成型的类库可以用之外，对于 Android 的开发者来说，激动人心的特性是现在可以进入到操作系统可以进入的地方。也就是说，如果要创建一个应用程序打一个电话，就可以调用手机的拨号界面，也可以创建一个应用程序来使用手机内部的 GPS。

谷歌已经非常迫切的奉送了一些特性：Android 的开发者可以将自己的应用程序和谷歌公司提供的如谷歌地图和谷歌搜索绑在一起。假设要写程序在谷歌地图上显示一个来电话者的位置，或者要储存一般的搜索结果到联系人中，在 Android 中，这个门已经完全打开。

1.2 开发环境的搭建

Android 应用程序是在 Java 下开发的。Android 本身不是一个语言，而是一个运行应用程序的环境。这样，理论上可以使用任何发布或综合开发环境（IDE）来开发。开放手机联盟和谷歌认同一个 Java 的 IDE，那就是 Eclipse。当然，Eclipse 也并非完美，由于 Eclipse

不是专为 Android 开发而设计的，因此存在很多缺点。谷歌公司在 2013 年的 I/O 大会上发布了 Android Studio——专为 Android 应用开发而设计的开发环境，该工具的开发环境和模式更丰富、便捷，能够支持多种语音，还可以为开发者提供测试工具和各种数据分析。由于该工具目前还是测试版（最新版本 0.2.x），因此，本书还是以传统的 Eclipse 为开发环境来介绍。

1.2.1 下载和安装 JRE

在下载和安装 Eclipse 之前，必须确保在电脑上下载并安装了 Java Runtime Environment（JRE，Java 运行时环境）。因为 Eclipse 作为一个程序是由 Java 写成，它依靠 JRE 来运行。如果 JRE 没有安装或被检测到，打开 Eclipse 时会看见错误提示。

大多数使用过网络或以网络为基础的应用程序的用户，应该安装过 JRE。JRE 允许在电脑上运行 Java 基础的应用程序，但是它不允许创建 Java 应用程序。要创建 Java 应用程序，需要下载并安装 Java Development Kit（JDK），这个包含了创建 Java 应用程序所需的所有工具和库。如果不熟悉 Java，记住这一点就行了。对于书中提到的例子，笔者会下载 JDK，因为它也包含了 JRE。

通过浏览器访问 Java 的下载页面（http://java.com/zh_CN/download/index.jsp），如图 1-1 所示。正常情况下只需要 JRE 来运行 Eclipse，但是对于本书来讲，应当下载包含了 JRE 的完整的 JDK。



图 1-1 JRE 下载页面

运行下载的 exe 文件，建议用户按照软件的默认设置来安装，以避免出现意外情况。

1.2.2 下载和安装 Eclipse

打开 Eclipse 官方网站（http://www.eclipse.org/downloads）的下载页面，如图 1-2 所示。

在这个站点下载为 Java 开发者准备的 Eclipse 的 IDE（Eclipse IDE for Java Developers）。不要下载 Eclipse IDE for Java EE 的开发包，因为这是不同的产品。

下载 Eclipse 以后，导航到软件包下载的位置。写本书时，最新的 Eclipse 软件包 Windows

版本的文件是 eclipse-java-kepler-R-win32.zip。解压缩软件包并且运行 Eclipse.exe。图 1-3 所示显示了软件启动的欢迎画面。

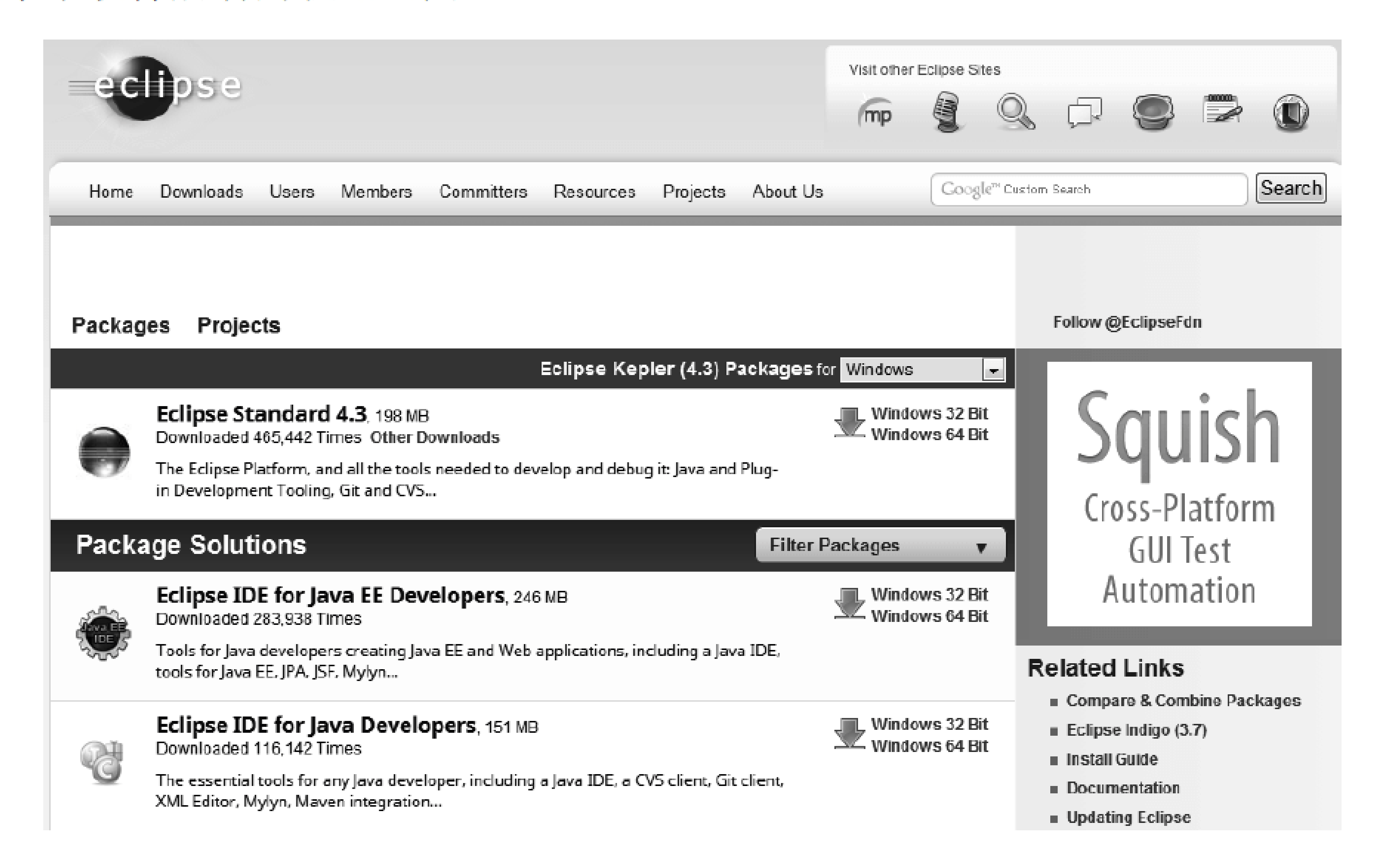


图 1-2　Eclipse 下载页面



图 1-3　Eclipse 启动界面



如果用户没有看见欢迎画面，试着重新启动电脑。如果重启后没有出现帮助窗口的话，只下载并安装 JRE。

第一次启动 Eclipse，会提醒用户创建一个缺省的工作空间或文件夹。和其他大多数开发环境相同，项目被创建，并且保存到该工作空间内。缺省的工作空间路径是用户路径，

也可以单击 Browse 选择不同路径，如图 1-4 所示。

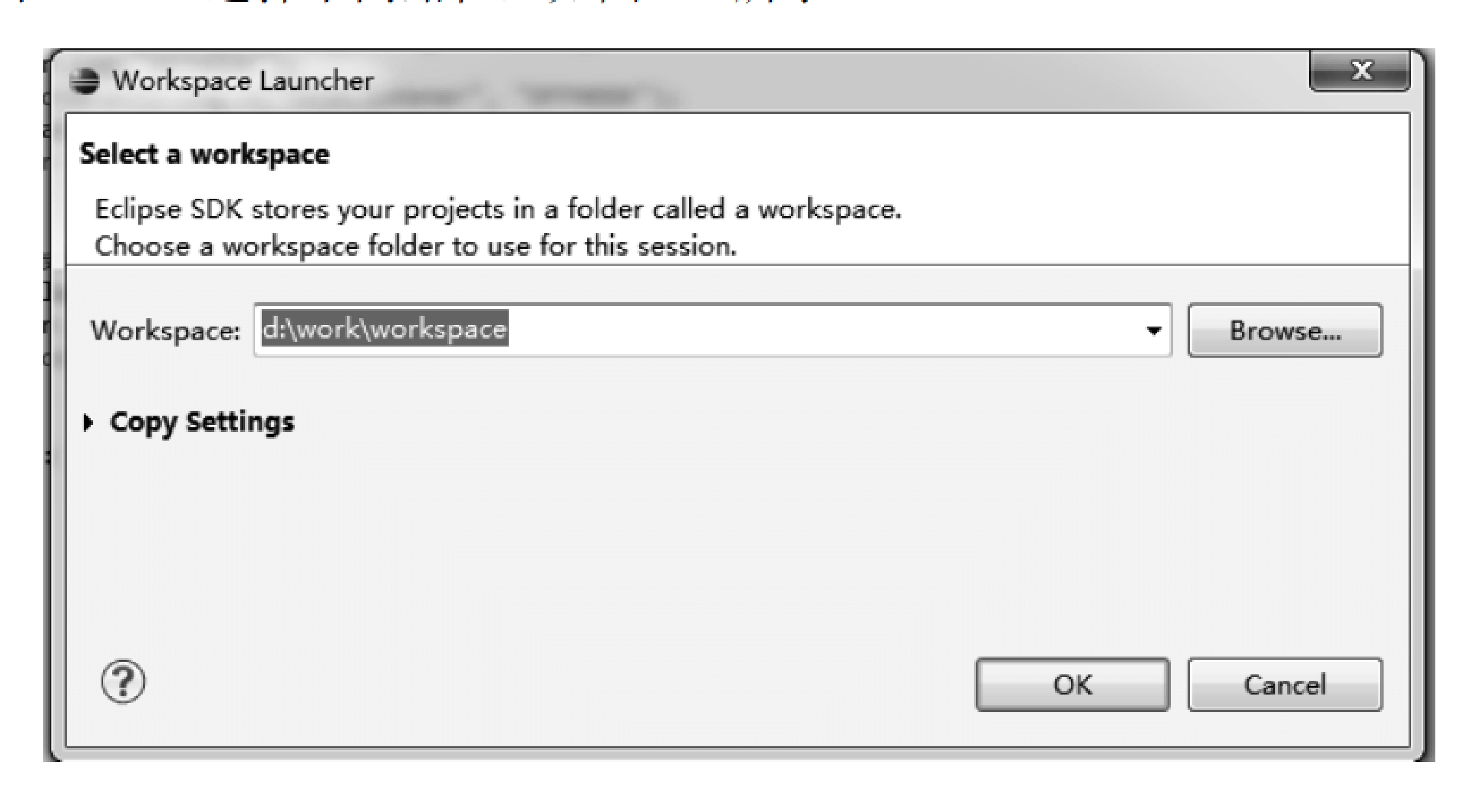


图 1-4　设置工作空间

建议用户指定一个固定的工作空间目录。这样，当创建新项目时，就会知道在哪个路径里能找到项目的源文件。在本书内，有时需要导航到项目文件，并且在 Android 开发环境的外部工作，所以知道文件的所在位置是非常有帮助的。选择工作空间之后，单击 OK。这样，开发环境就安装好了。

1.2.3　Android SDK 和 ADT

原先搭建 Android 开发环境时，需要分别下载 Eclipse、Android SDK 和 ADT（Android Developer Tools），现在，谷歌已经将三者集成在了一起，无需再分别下载配置了。

用浏览器访问 Android 开发者网站（http://developer.android.com/sdk/index.html），如图 1-5 所示。单击“Download the SDK”下载。

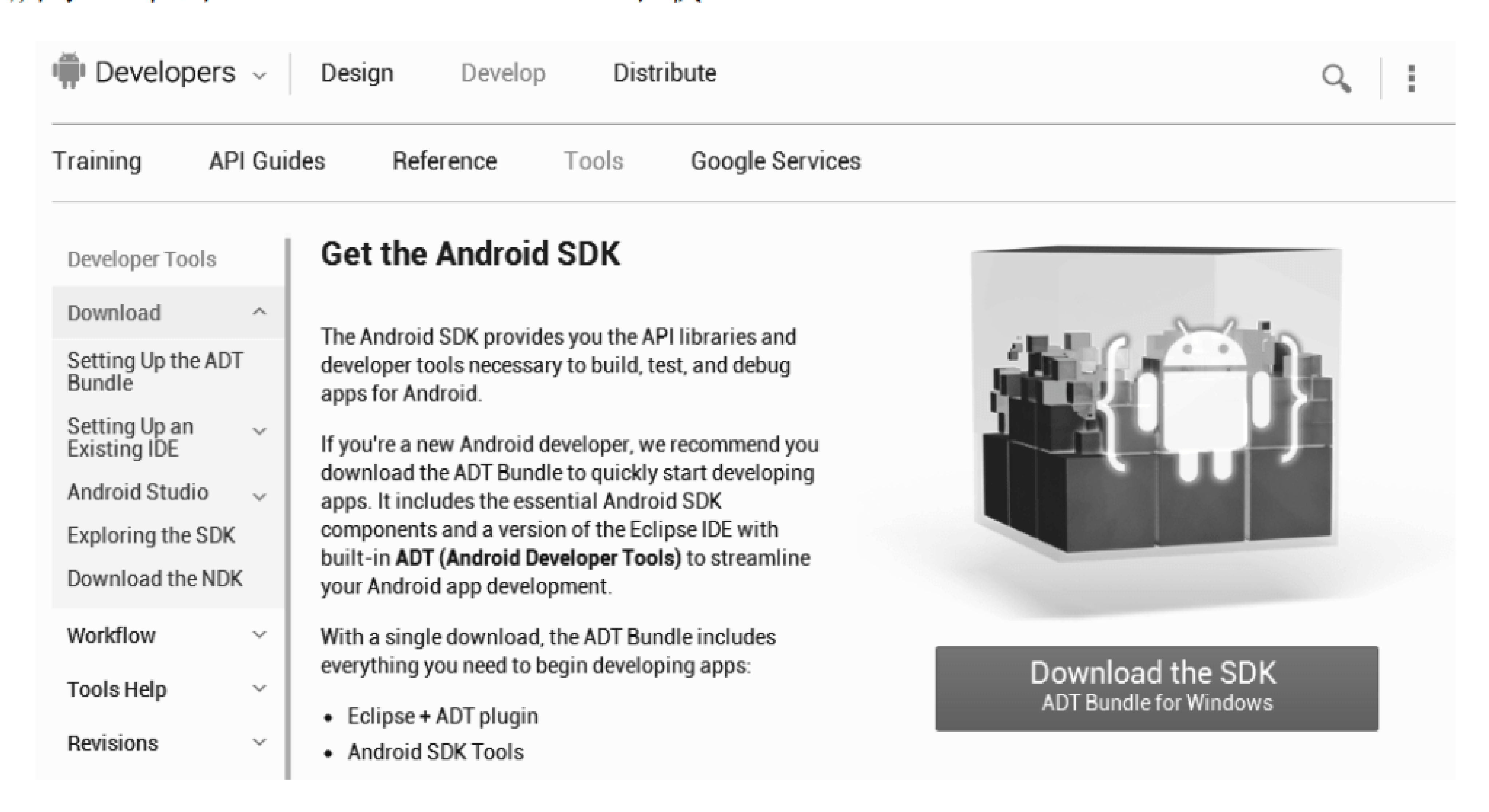


图 1-5　集成开发环境下载

将下载的文件（目前最新版本为 adt-bundle-windows-x86-20130522.zip）解压缩，压缩包中包括 Eclipse、最新版的 SDK 和 ADT，直接启动 Eclipse 即可。

1.2.4　管理 SDK 和 AVD

在下载的集成开发环境中，只包含最新版本的 Android SDK（目前为 4.2 版），如果要开发其他版本的 Android 应用程序还需通过“Android SDK Manager”程序联网下载。同时，应用程序的调试需要虚拟机（AVD——Android Virtual Device）来运行，因此，开发者必须掌握“Android Virtual Device Manager”程序的使用。

启动集成环境中的 Eclipse，单击如图 1-6 所示圈出的图标，启动 Android SDK Manager 程序。

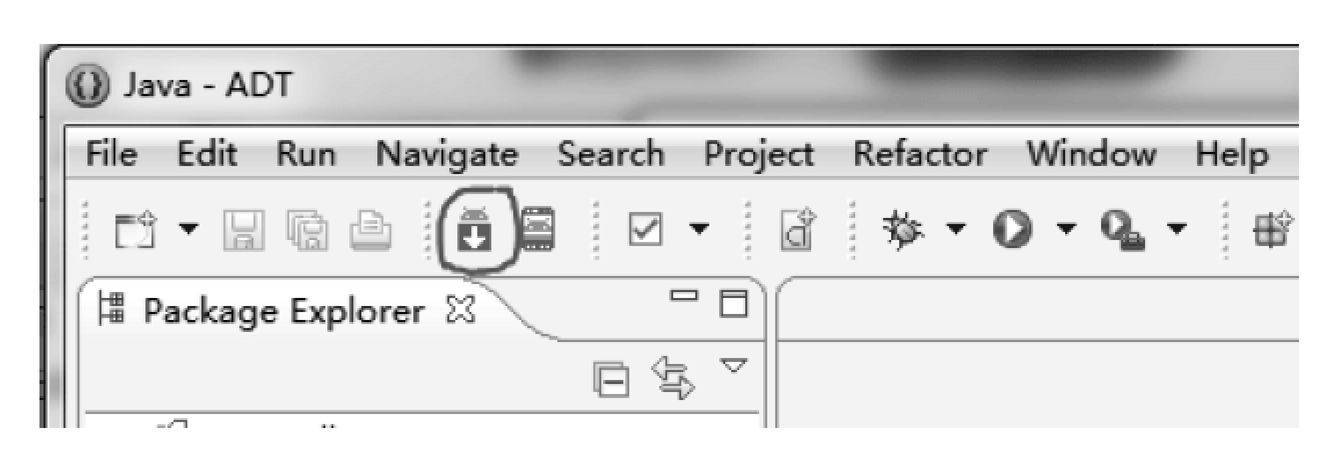


图 1-6　启动 Android SDK Manager 程序

SDK 管理程序如图 1-7 所示，通过该程序可以管理开发所需的各种工具和不同版本的 SDK。选择需要的包（Packages），单击“Install”按钮即可。

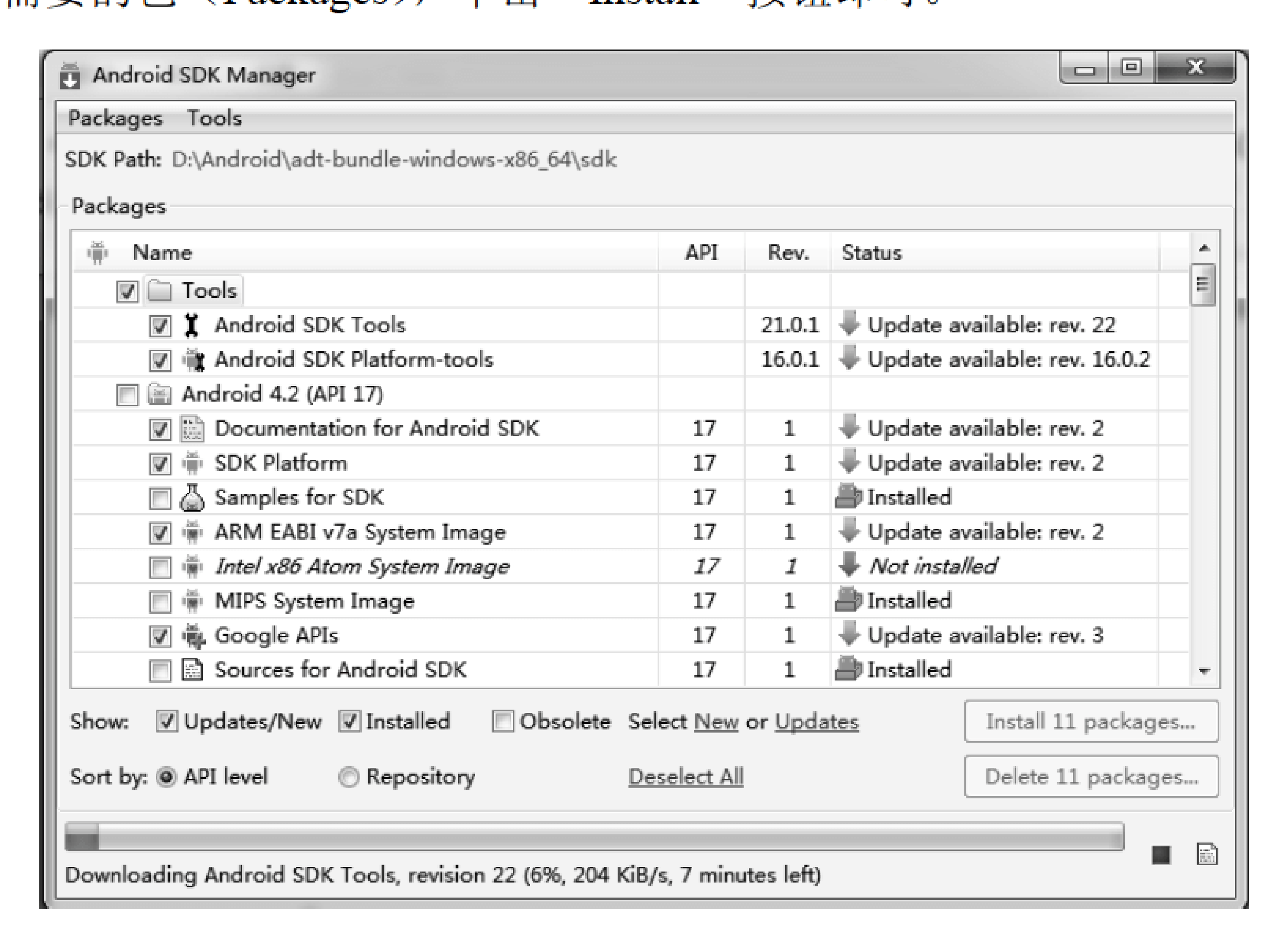


图 1-7　Android SDK Manager 界面

Android 虚拟机管理是经常会用到的功能，单击“Android SDK Manager”旁边的“Android Virtual Device Manager”按钮，即可启动虚拟机管理程序，如图 1-8 所示。

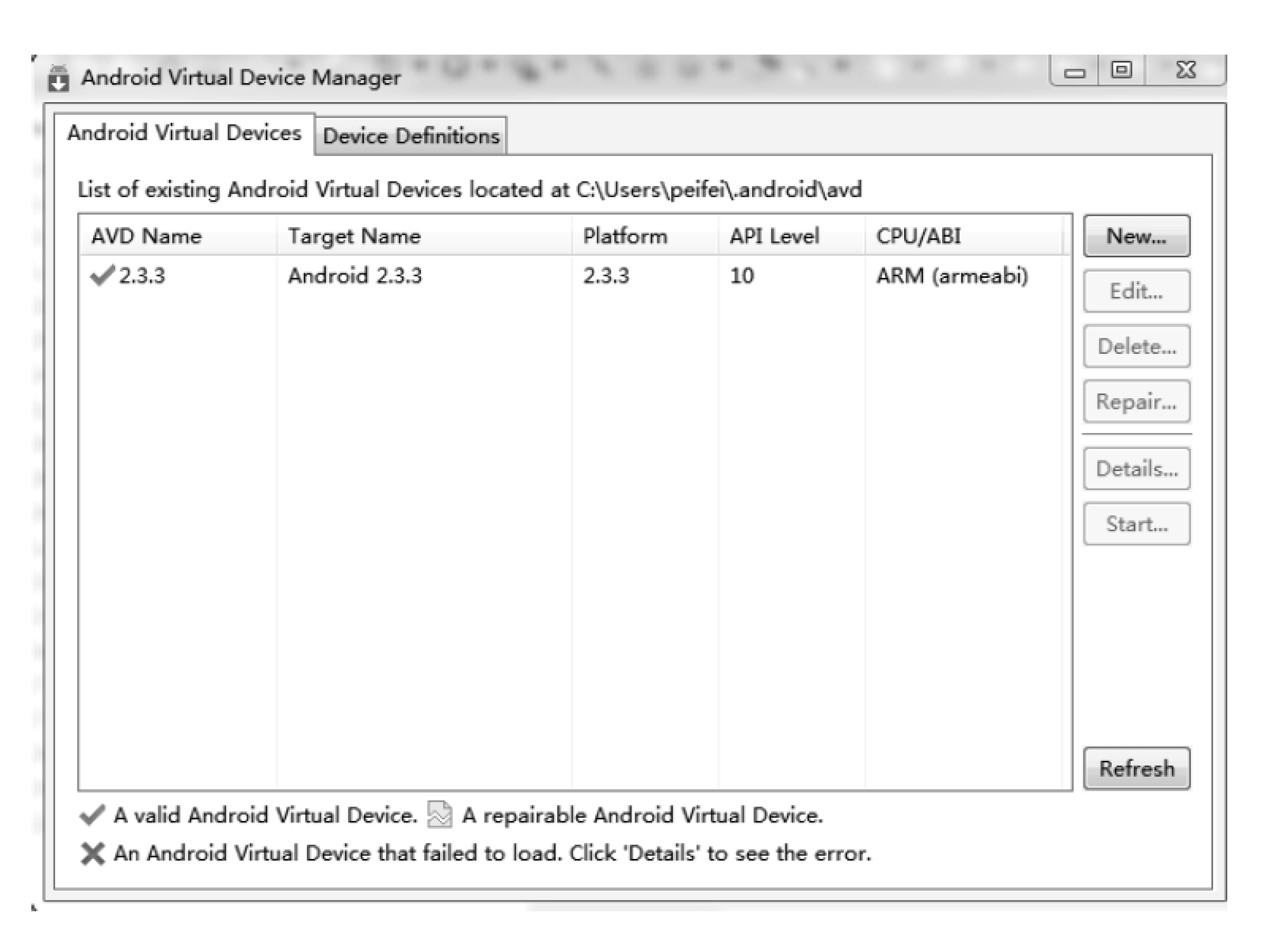


图 1-8 AVD Manager 界面

单击“New…”按钮，创建一个虚拟机，如图 1-9 所示。

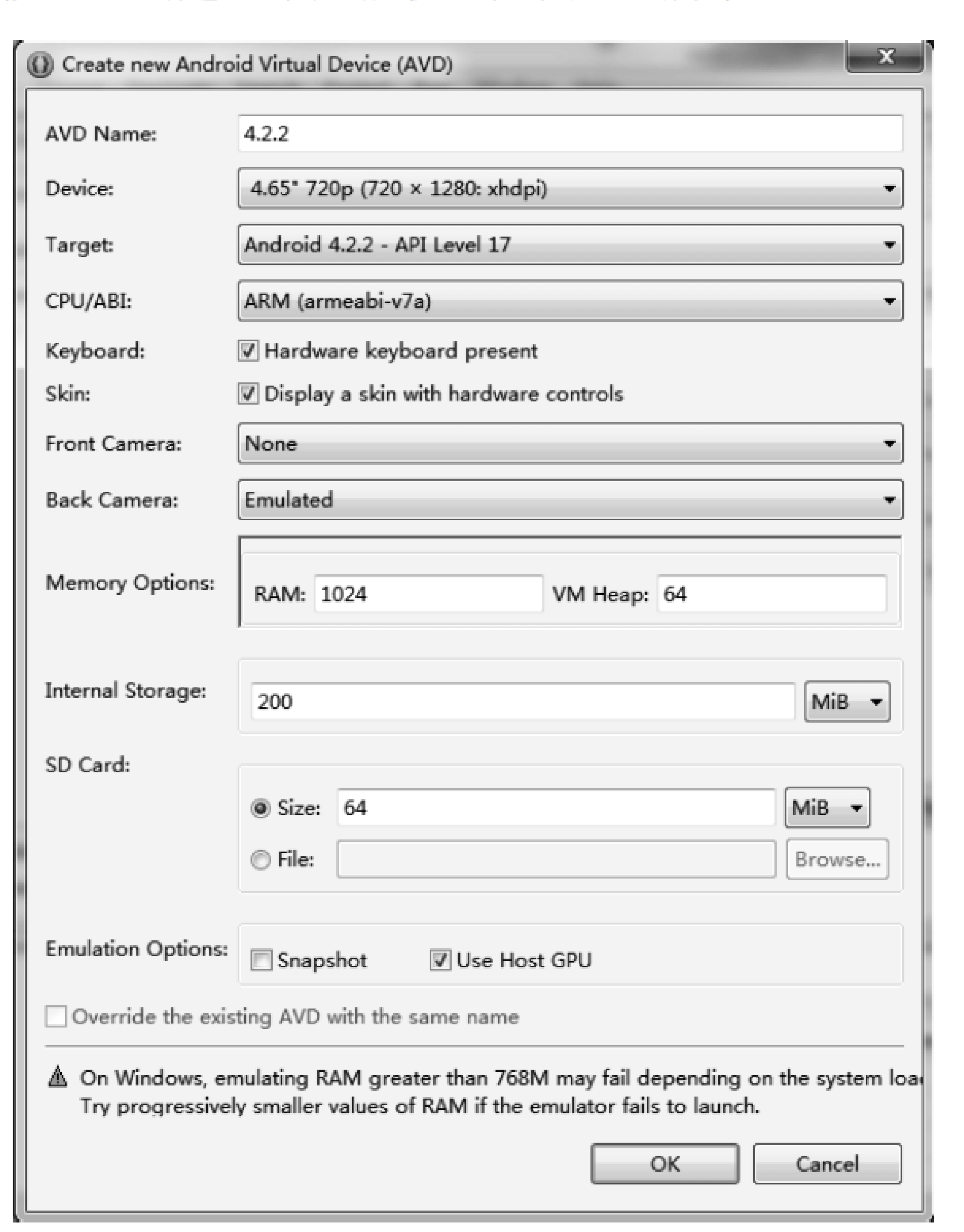


图 1-9 创建虚拟机

各选择含义如下。

- **AVD Name** 虚拟机名称，建议用 SDK 版本号命名，以便识别。

- **Device**　虚拟机屏幕尺寸，根据需要选择，建议用当前主流设备的屏幕尺寸。
- **Target**　SDK 版本号，根据需要选择。
- **CPU/ABI**　CUP 类型，选择“ARM”。
- **Keyboard**　是否带有实体键盘。
- **Skin**　是否显示实体外观。
- **Front Camera**　前置摄像头。
- **Back Camera**　后置摄像头。
- **Memory Options**　内存选项。
- **Internal Storage**　内部存储。
- **SD Card**　SD 卡容量。
- **Emulation Options**　虚拟化选项。

创建成功后，单击“Start…”按钮即可启动虚拟机，如图 1-10 所示，今后在开发应用程序时，即可在虚拟机中调试运行。

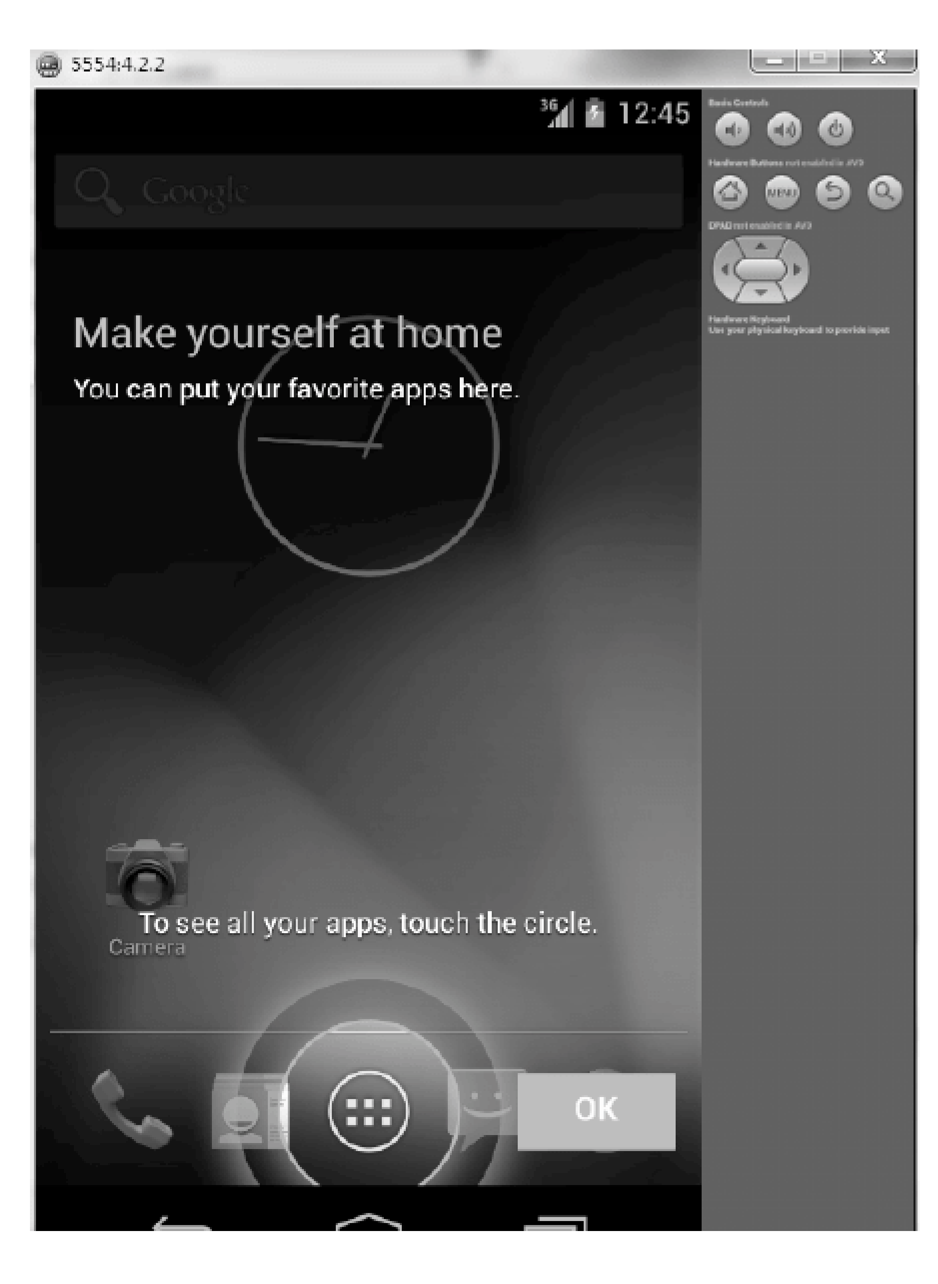


图 1-10　虚拟机启动界面

1.3　创建第一个 android 应用程序

下面创建一个“Welcome　Android!”的应用程序，从高级层面上有 3 个步骤。

- 通过选择 File→New→Project 菜单，建立新项目 “Android Project”。
- 填写新项目各种参数。
- 编辑自动生成的代码模板。

详细的步骤如下。

（1）打开 Eclipse，新建项目（单击 File→New→Project 菜单），在项目列表中展开 Android 目录，选择 Android Application Project，如图 1-11 所示。

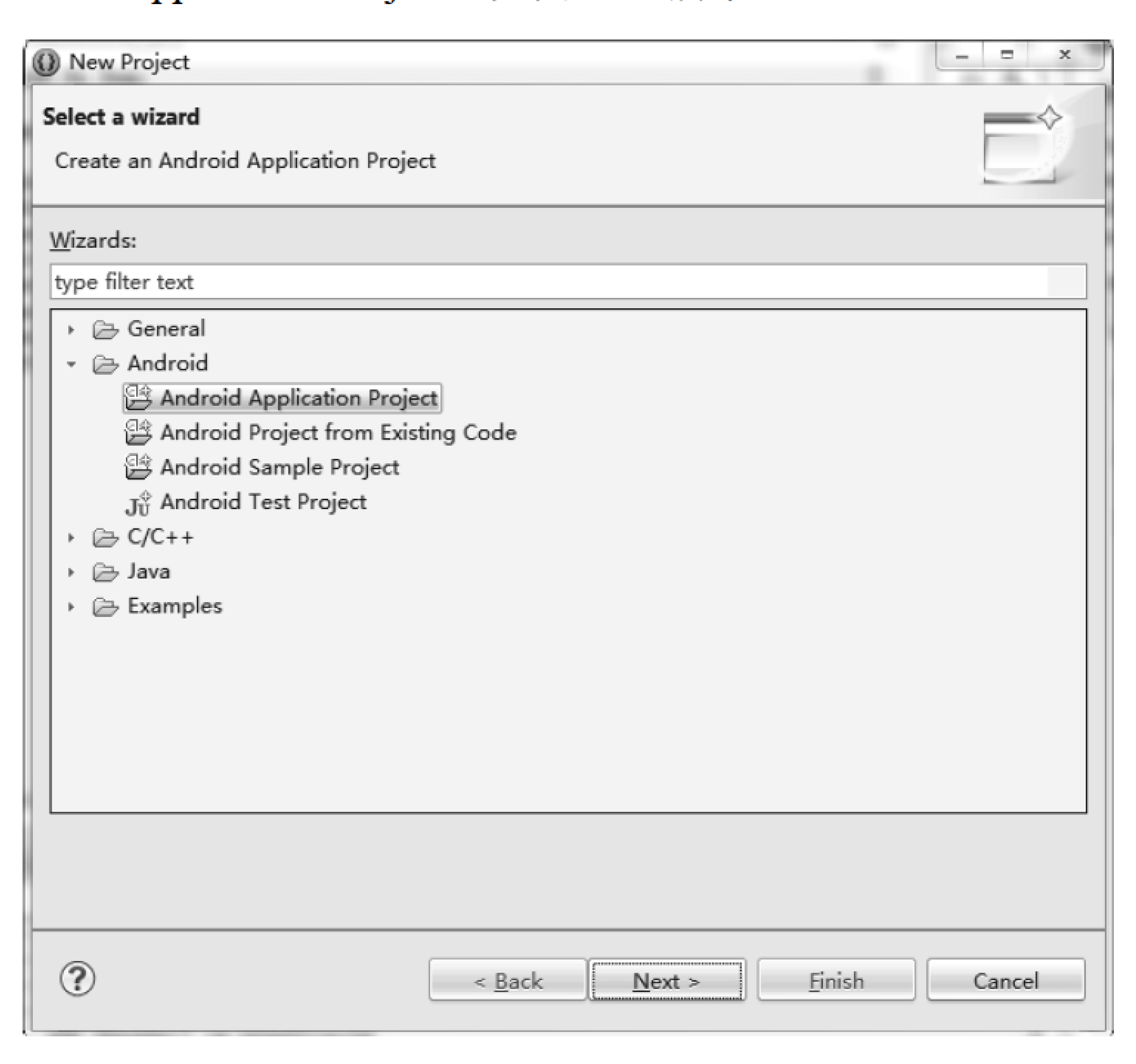


图 1-11 “New Project”对话框

（2）单击“Next”按钮，弹出“New Android Application”对话框，在此对话框填写项目的细节参数。本案例填写完后的对话框如图 1-12 所示。

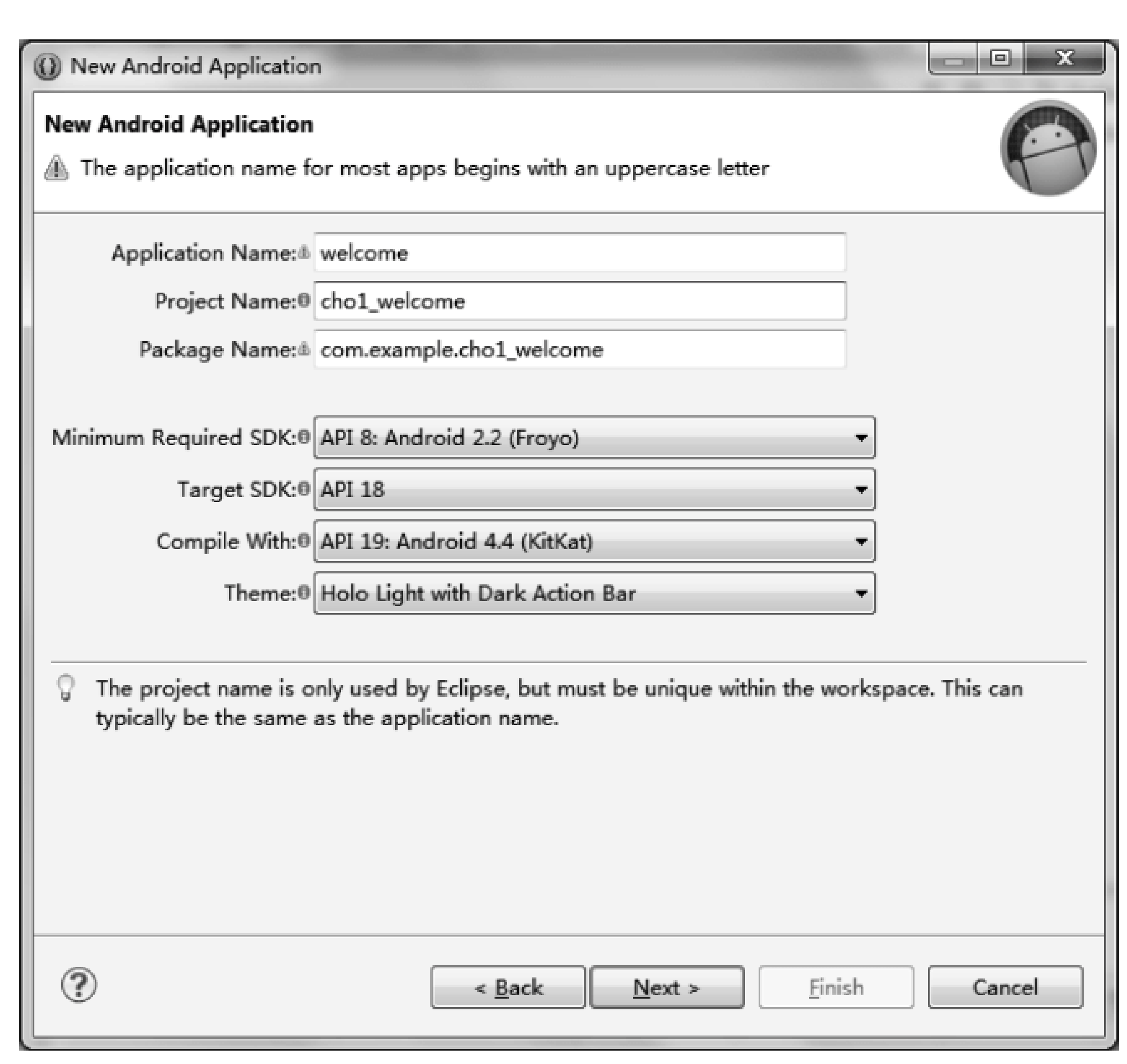


图 1-12 “New Android Application”对话框

各个参数的含义如下。

- **Application Name**　一个易读的标题出现在应用程序上。在“选择栏”的“Use default location”选项，允许用户选择一个已存在的项目。
- **Project Name**　包含这个项目的文件夹的名称。
- **Package Name**　包名，遵循 Java 规范，用包名来区分不同的类是很重要的，例子中用到的是“com.example.cho1-welcome”，用户可以按照自己的计划命名一个有别于该路径的名称。

（3）单击“Next”按钮，弹出如图 1-13 所示的“Configure Launcher Icon”对话框。

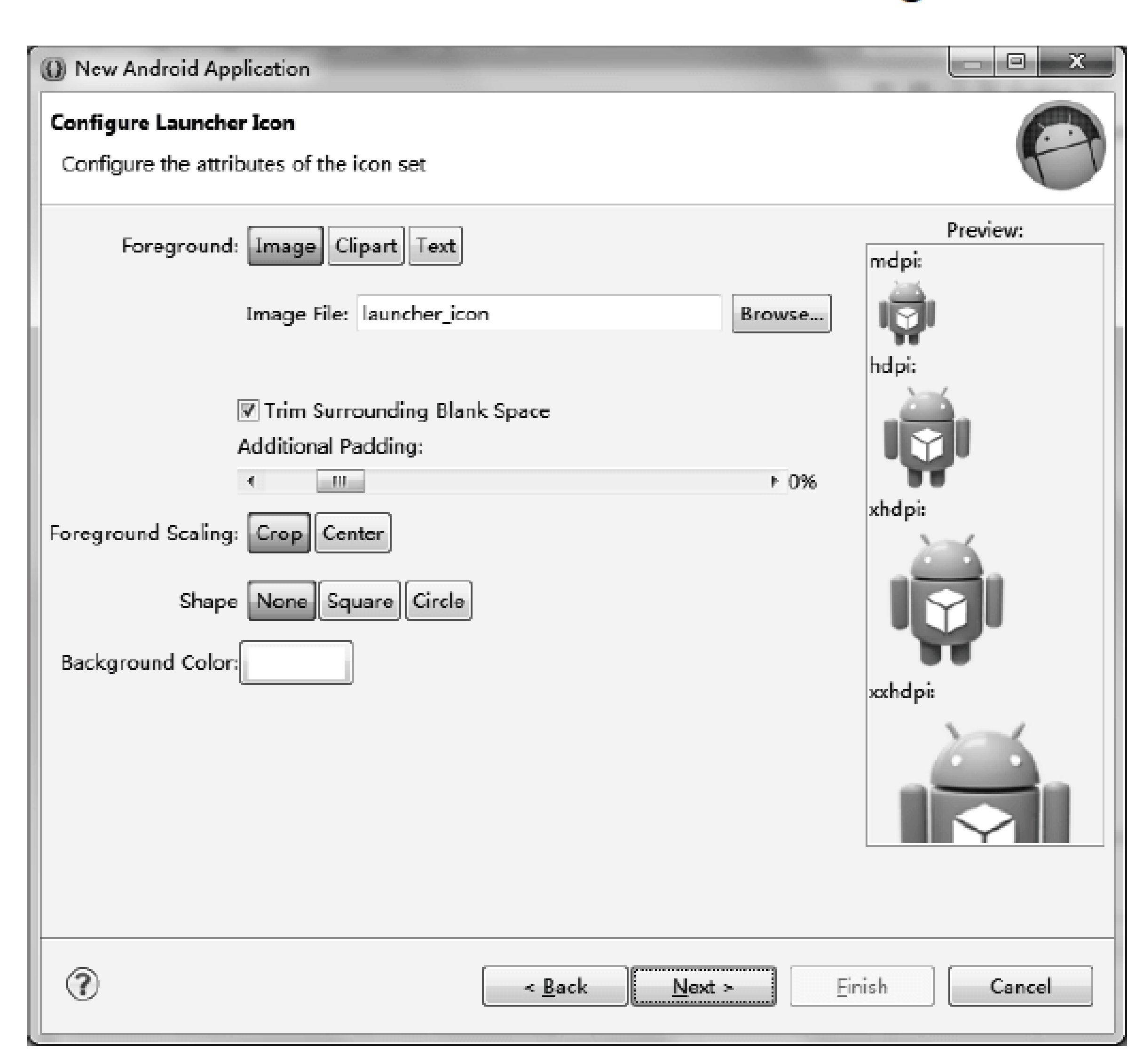


图 1-13　“Configure Launcher Icon”对话框

（4）单击“Next”按钮，弹出如图 1-14 所示的“Create Activity”对话框。



图 1-14　“Create Activity”对话框

（5）单击“Next”按钮，弹出如图 1-15 所示的“Blank Activity”对话框。

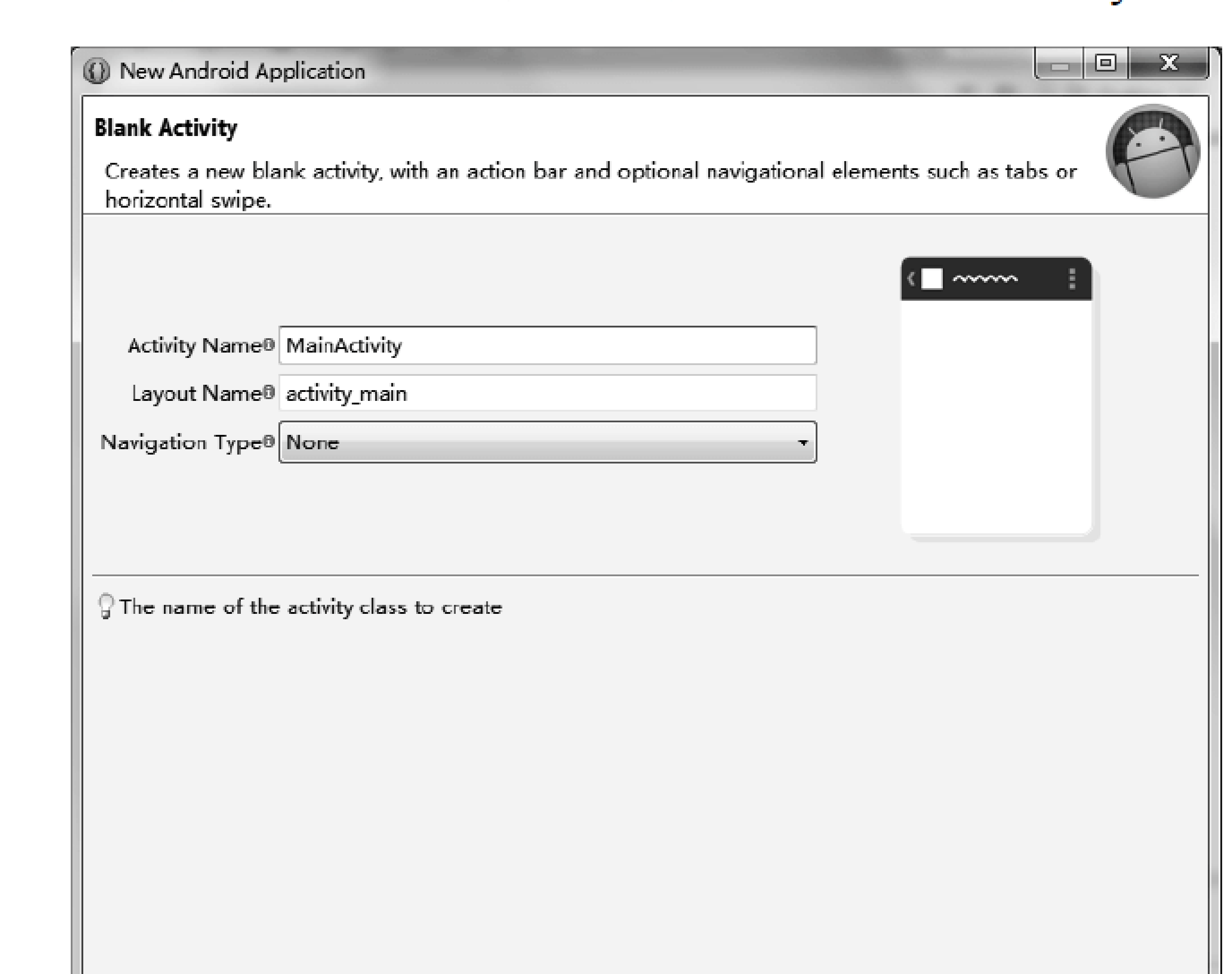


图 1-15 “Blank Activity”对话框

参数的含义如下。

- ❑ **Activity Name** 项目的主类名，该类将会是 Android 的 Activity 类的子类。一个 Activity 类是一个简单的启动程序和控制程序的类，它可以根据需要创建界面，但不是必须的。
- ❑ **Layout Name** 布局的名称。

（6）单击“Finish”按钮，出现如图 1-16 所示的程序界面。

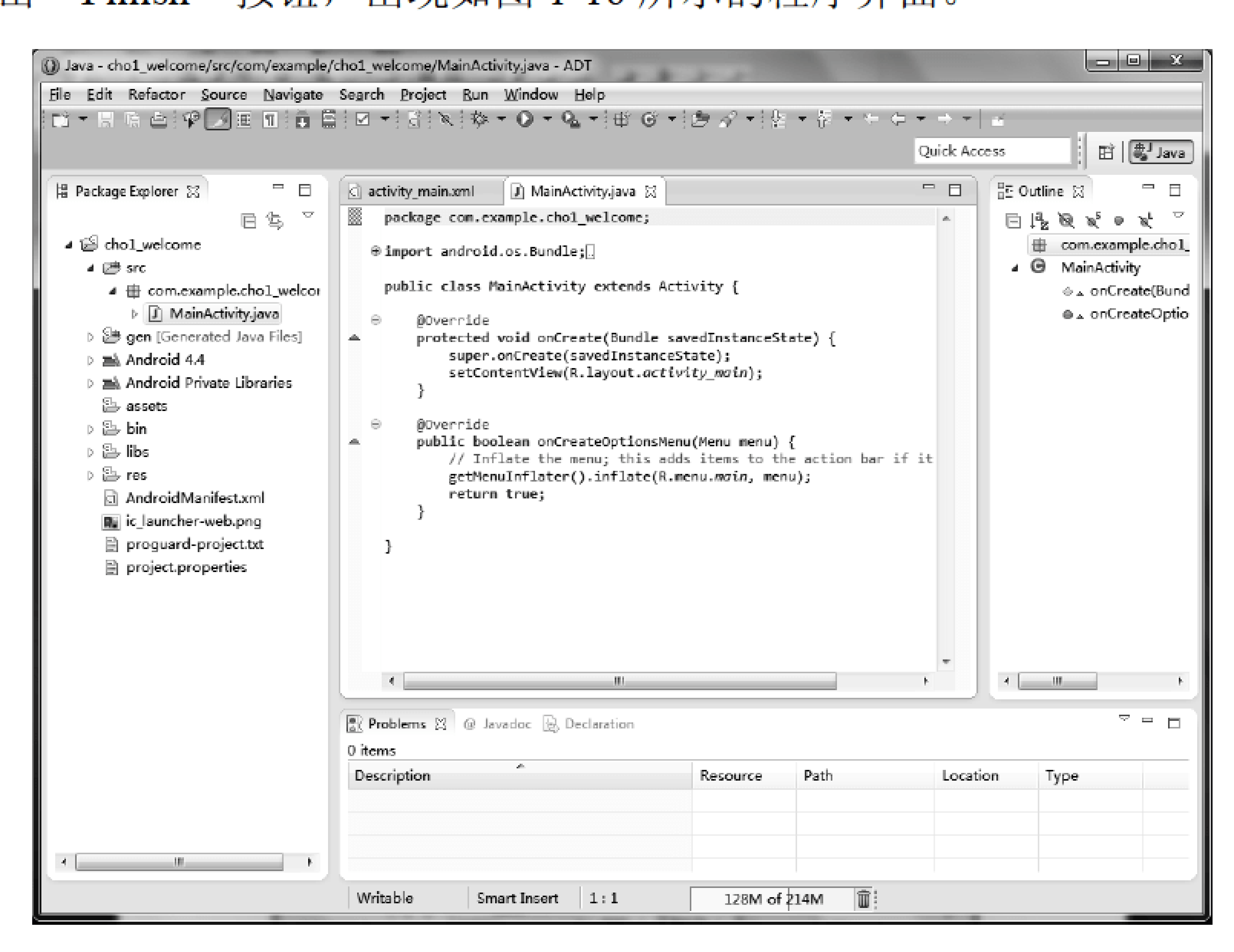
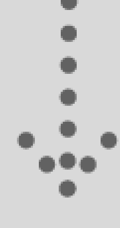


图 1-16 程序界面





（7）修改 res/values/strings.xml 中的文件，如图 1-17 所示。

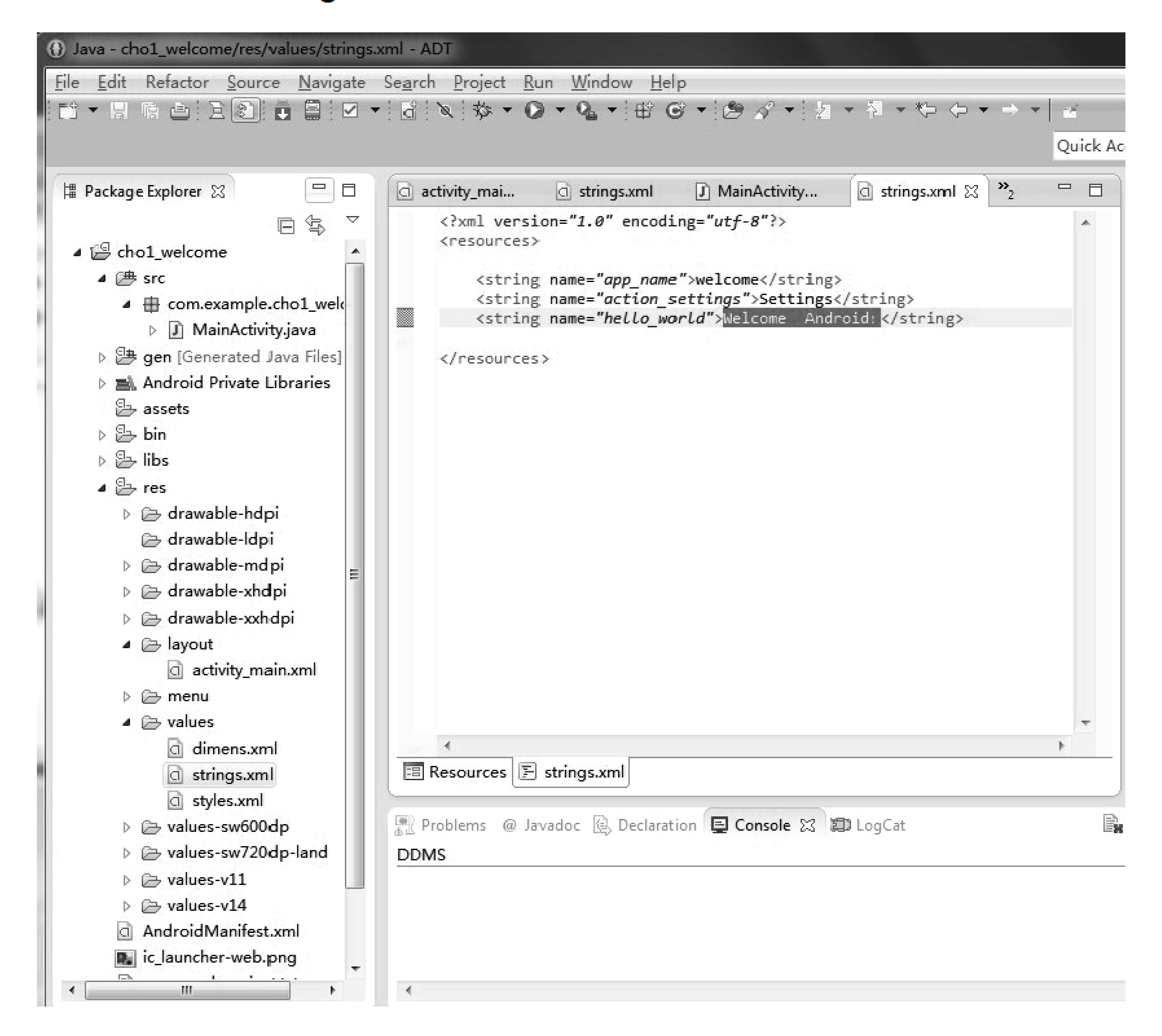


图 1-17　修改 strings.xml 文件内容

（8）运行程序，右击“cho1_welcome”，在弹出的快捷菜单中选择 Run As→Android Application 选项，如图 1-18 所示。

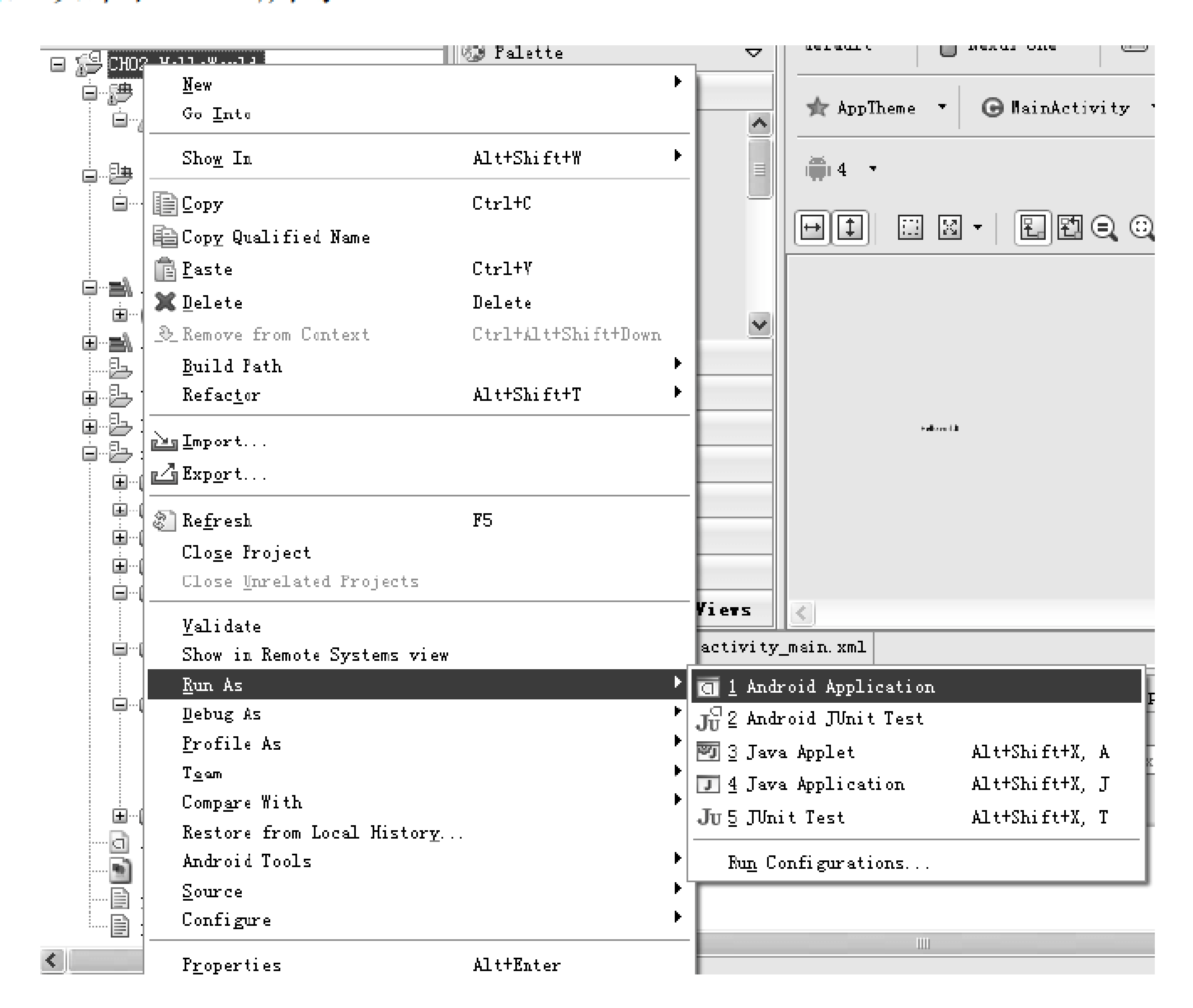


图 1-18　运行步骤

（9）程序运行结果如图 1-19 所示。

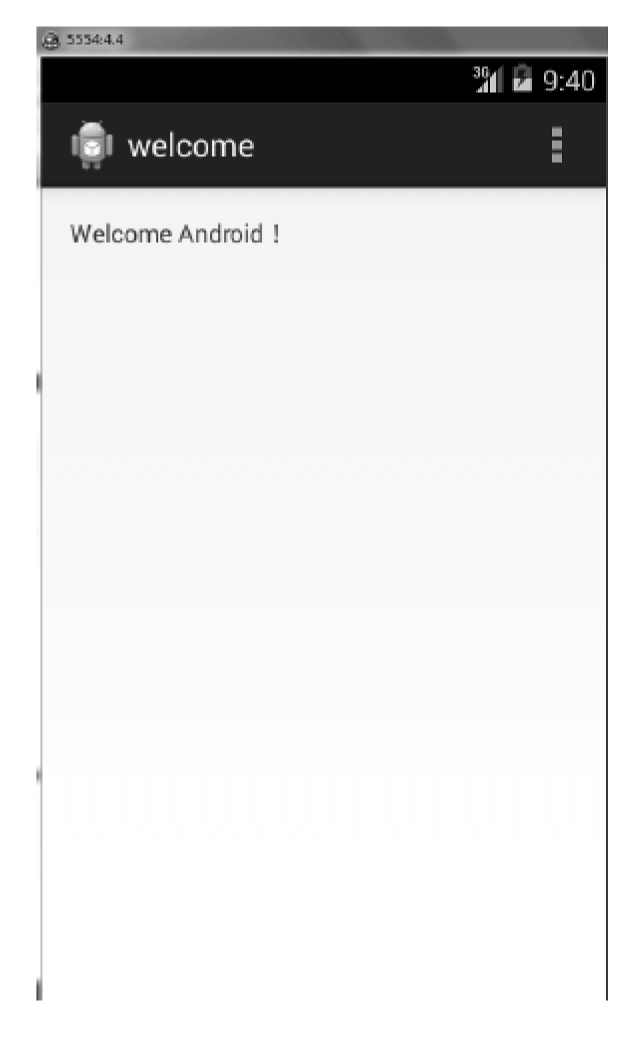


图 1-19　程序运行结果

1.4　Android 系统架构及应用程序的结构

读者也许会有疑问，1.3 节中只修改了 res/values/strings.xml 文件中的一小部分，其他文件都没修改，程序就运行出如图 1-19 所示的界面。那么 Android 系统到底是怎样运行的呢？下面对 Android 系统架构及应用程序的结构进行分析。

1.4.1　Android 系统架构

Android 的系统架构和其操作系统一样，采用分层架构的思想，从上层到下层分别是应用程序层、应用程序框架层、系统运行库层及 Linux 内核层，如图 1-20 所示。

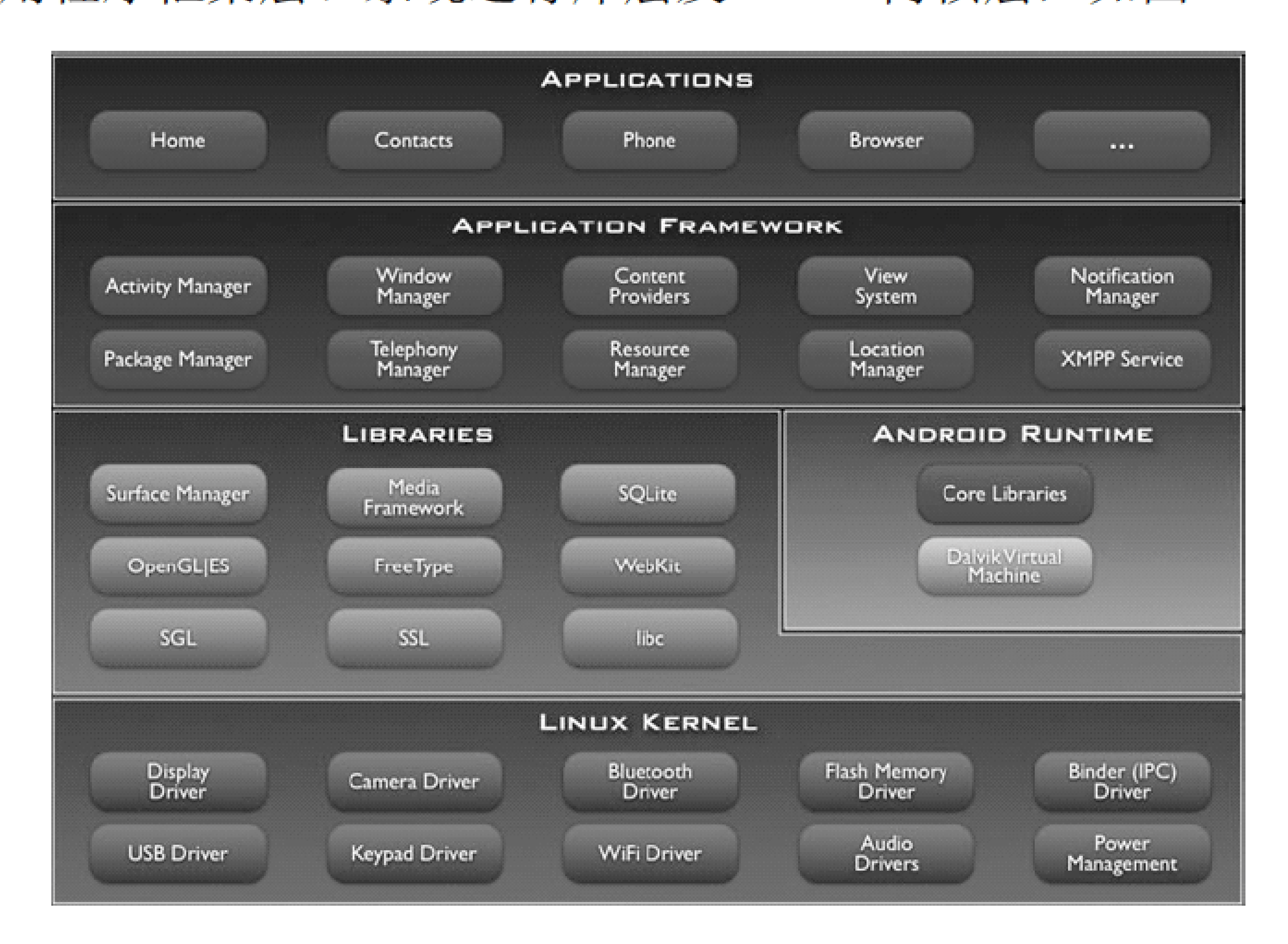
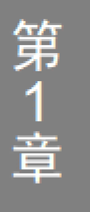


图 1-20　Android 的系统架构



每一层的作用如下。

1．应用程序层（Applications）

这一层提供一些核心应用程序包，如电子邮件、短信、电话拨号程序、地图、图片浏览器、Web 浏览器和联系人管理等。这些应用程序都是开发人员所编写的，而所用到的类都是调用 Application Framework 层中的类库，并且这些应用程序都是可以被开发人员开发的其他应用程序所替换，这点不同于其他手机操作系统固化在系统内部的系统软件，更加灵活和个性化。

2．应用程序框架层（Application FrameWork）

这一层是 Android 应用开发的基础，提供了一些手机开发的最基本的 API，开发人员在开发应用程序时就是基于这层开发的，该层包括活动管理器、窗口管理器、内容提供者、视图系统、包管理器、电话管理器、资源管理器、位置管理器、通知管理器和 XMPP 服务十个部分，都是用 Java 开发的。

3．系统运行库层

包含函数库层（Libraries）和执行层（Android Runtime），函数库层是程序包，包括图层管理、媒体库、SQLite、OpenGLEState、FreeType、WebKit、SGL、SSL 和 libc，一般是用 C 或 C++编写的。执行层是 Android 运行环境，包括核心包与 google 自己开发的针对手机设备优化过的虚拟器。

4．系统核心层（Linux Kernel）

Android 操作系统都是基于 Linux 核心，其核心系统服务如安全性、内存管理、进程管理、网路协议及驱动模型都依赖于 Linux 内核，包括显示器驱动程序、照相机驱动程序、电源管理显示驱动、摄像头驱动、键盘驱动、WiFi 驱动、Audio 驱动、Flash 内存驱动、Binder（IPC）驱动等。Linux 提供的是最核心最基础的一些功能。

1.4.2　应用程序的项目结构

从 1.3 节的例子看出，通过 Android SDK 可以自动生成一个项目包，但是没有对项目包里的内容进行介绍，本小节对项目包中的内容进行一一介绍。展开“Package Explorer”窗口中的“ch01_welcome”项目名称，看到如图 1-21 所示的目录结构。

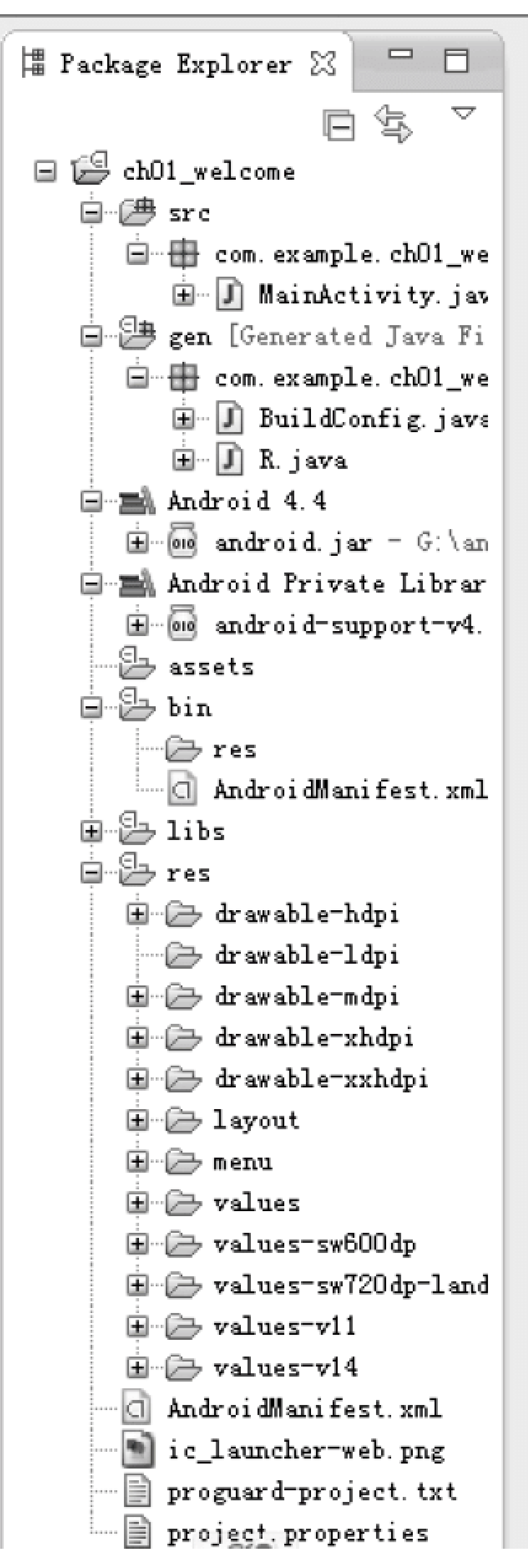


图 1-21　目录结构

1．src 源代码目录

该目录存放 Android 应用程序所有的源代码，里面一般都是.java 结尾的 java 文件，该目录项有不同的包，包中对应开发的源程序，开发的主要精力都集中在开发 src 目录下的内容。打开 MainActivity.java，代码清单如下。

代码清单：src/com.example.ch01_welcome/MainActivity.java

```
package com.example.ch01 welcome;          //声明包名
import android.os.Bundle;                  //引入 Bundle 包
import android.app.Activity;               //引入 Activity 包
import android.view.Menu;
public class MainActivity extends Activity {//从 Activity 类派生子类 MainActivity
    @Override
    protected void onCreate(Bundle savedInstanceState) { //重写 onCreate 方法
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity main);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }
}
```

2. gen 文件夹目录

该目录存放了 Android 开发工具自动生成的文件。目录中有个包名，该包名是自己定义的。在包中有两个文件；一个是 BuildConfig.java 文件；另一个是 R.java 文件。BuildConfig.java 文件是 Android 调试用的。R.java 文件才是最重要的，定义了一个 R 类，它包含了应用中用户界面、图像、字符串等各种资源与之相对应的资源编号（id），这些资源编号根据 res 目录的资源是系统自动生成的，即有一个资源对象，系统就为此在 R 类中生成相应的资源编号；R.java 文件在 Application 中起到字典的作用，它包含了各种资源的地址（ID），通过 R.java 文件，用户可以方便找到相应的资源元素。BuildConfig.java 和 R.java 文件最好不要手动去修改。打开 R.java 文件，代码清单如下。

代码清单：gen/com.example.ch01_welcome/R.java

```
/* AUTO-GENERATED FILE.  DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found.  It
 * should not be modified by hand.
 */

package com.example.ch01 welcome;
public final class R {
    public static final class attr {
    }
    public static final class dimen {
        /**  Default screen margins, per the Android Design guidelines.

         Customize dimensions originally defined in res/values/dimens.xml (such as
         screen margins) for sw720dp devices (e.g. 10" tablets) in landscape here.
            */
        public static final int activity horizontal margin=0x7f040000;
        public static final int activity_vertical_margin=0x7f040001;
```

```
    }
    public static final class drawable {
        public static final int ic_launcher=0x7f020000;
    }
    public static final class id {
        public static final int action_settings=0x7f080000;
    }
    public static final class layout {
        public static final int activity_main=0x7f030000;
    }
    public static final class menu {
        public static final int main=0x7f070000;
    }
    public static final class string {
        public static final int action_settings=0x7f050001;
        public static final int app_name=0x7f050000;
        public static final int hello_world=0x7f050002;
    }
    public static final class style {
        /**
        Base application theme, dependent on API level. This theme is replaced
        by AppBaseTheme from res/values-vXX/styles.xml on newer devices.
           Theme customizations available in newer API levels can go in
            res/values-vXX/styles.xml, while customizations related to
            backward-compatibility can go here.
        Base application theme for API 11+. This theme completely replaces
        AppBaseTheme from res/values/styles.xml on API 11+ devices.
 API 11 theme customizations can go here.
        Base application theme for API 14+. This theme completely replaces
        AppBaseTheme from BOTH res/values/styles.xml and
        res/values-v11/styles.xml on API 14+ devices.

 API 14 theme customizations can go here.
         */
        public static final int AppBaseTheme=0x7f060000;
        /**  Application theme.
 All customizations that are NOT specific to a particular API-level can go here.
         */
        public static final int AppTheme=0x7f060001;
    }
}
```

程序清单中定义了一些常量，但是名字与 res 文件夹中的文件名相同，证明 R.java 文件所存储的是该项目所有资源的索引，最好不要手动去修改。

3．Android 4.4

这是 Android 提供的一个架文件，用户所引用的所有类都来源于该架文件。

4．res 资源目录

存放使用到的各种资源，如 XML 界面文件、图片、数据等。res/drawable 开头的五个目录，有 drawable-ldpi、drawable-mdpi、drawable-hdpi、drawable-xhdpi 和 drawable-xxhdpi，

主要为了支持多分辨率，例如，

（1）drawable-hdpi 里存放高分辨率的图片，如 WVGA (480x800),FWVGA (480x854)。

（2）drawable-mdpi 里存放中等分辨率的图片，如 HVGA (320x480)。

（3）drawable-ldpi 里存放低分辨率的图片，如 QVGA (240x320)。

系统会根据机器的分辨率来分别到这几个文件夹中寻找对应的图片，开发人员可以通过 Resource.getDrawable(id)获得该资源。

res/layout 目录专门存放 XML 界面文件，主要用于表述应用程序的用户界面布局，也用于描述用户界面和接口组件。一般一个用户界面布局一个 XML 文件。打开 activity_main.xml 文件，代码清单如下。

代码清单：res/layout/activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout width="match parent"
   android:layout height="match parent"
   android:paddingBottom="@dimen/activity vertical margin"
   android:paddingLeft="@dimen/activity horizontal margin"
   android:paddingRight="@dimen/activity horizontal margin"
   android:paddingTop="@dimen/activity vertical margin"
   tools:context=".MainActivity" >

   <TextView
      android:layout_width="wrap_content"
      android:layout height="wrap content"
      android:text="@string/hello world" />

</RelativeLayout>
```

res/values 目录专门存放使用到的各种类型的资源，不同类型的资源存放在不同的使用 XML 格式的参数描述的文件中，如 string.xml 定义字符串和数值，style.xml 定义了样式，dimens.xml 定义了资源的重用 。开发者也可以在此添加一些额外的资源如颜色（color.xml）和数组（arrays.xml）等。主要用于在代码中通过 R 类来调用它们，而不直接使用，其作用是将代码和资源分开管理，便于维护。打开 strings.xml 文件，代码清单如下。

代码清单：res/values/strings.xml

```

<?xml version="1.0" encoding="utf-8"?>
<resources>

   <string name="app name">welcome</string>
   <string name="action settings">Settings</string>
   <string name="hello world">Welcome Android!</string>

</resources>
```

strings.xml 文件中定义文件中所要用到的一些常量，代码清单中定义了 3 个字符串及值，“app_name”字符串对应的值是“welcome”，“action_settings”字符串对应的值是

“Settings”，“hello_world”字符串对应的值是“Welcome Android!”。这些字符串值一般通过对应的字符串引用显示在界面上，如果要把这些 Android 英文版修改成中文版，一般情况下只要把 strings.xml 中字符串值该成相对应的中文即可。

5. assets 资源目录

一般用于存放 HTML 文件、数据库文件、JavaScript 文件，assert 目录下的文件不会在 R.java 自动生成 ID，所以读取 assets 目录下的文件必须指定文件的路径。



res 和 assets 都是放置文件，但是这两个最主要的区别是 res 目录下的文件在 R.java 自动生成 ID，但是，assert 目录下的文件不会在 R.java 自动生成 ID。

6. AndroidManifest.xml 项目清单文件

该文件列出了应用程序提供的功能，开发好的各种组件需要在此文件中进行配置，尤其是 Activity、Intent、Service 及 ContentProvider，凡是需要用到的组件都要在此注册，当使用到系统内置的应用（如电话服务、互联网服务、短信服务、GPS 服务等）时，还需在此文件中声明使用权限，该文件也是所有 Android 应用程序都需要的文件，描述了程序包的全局变量，包括公开的应用程序组件和每个组件的实现类，什么样的数据可以操作，在什么地方可以运行等。打开 AndroidManifest.xml 文件，代码清单如下。

代码清单：AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.ch01 welcome"
   android:versionCode="1"
   android:versionName="1.0" >

   <uses-sdk
      android:minSdkVersion="8"
      android:targetSdkVersion="18" />

   <application
      android:allowBackup="true"
      android:icon="@drawable/ic launcher"
      android:label="@string/app name"
      android:theme="@style/AppTheme" >
      <activity
         android:name="com.example.ch01 welcome.MainActivity"
         android:label="@string/app name" >
         <intent-filter>
             <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>
   </application>
</manifest>
```

分析 AndroidManifest.xml 文件的标签作用。

- ❑ **manifest**　根节点，描述了 package 中所有的内容。
- ❑ **uses-permission**　请求此 package 正常运作所需赋予的安全许可。
- ❑ **permission**　声明了安全许可来限制哪些程序可以使用该 package 中的组件和功能。
- ❑ **instrumentation**　声明了用来测试此 package 或其他 package 指令组件的代码。
- ❑ **application**　包含 package 中 application 级别组件声明的根节点。
- ❑ **activity**　Activity 是用来与用户交互的主要工具。
- ❑ **receiver**　IntentReceiver 能使 application 获得数据的改变或发生的操作，即使该 application 当前不在运行。
- ❑ **service**　Service 是能在后台运行任意时间的组件。
- ❑ **provider**　ContentProvider 用来管理持久化数据并发布给其他应用程序使用的组件。

1.5　Android 应用程序组件

在 Android 程序中没有入口点（Main 方法），取而代之的是一系列的应用程序组件，这些组件都可以单独实例化。本节介绍 Android 支持的 4 种应用组件的基本概念。应用程序对外共享功能一般也是通过这 4 种应用程序组件实现的。

1.5.1　Activity（Android 的窗体）

Activity 是 Android 的核心类，该类的全名是 android.app.Activity。Activity 相当于 C/S 程序中的窗体（Form）或 Web 程序的页面。每一个 Activity 提供了一个可视化的区域。在这个区域可以放置各种 Android 控件，如按钮、图像、文本框等。

在 Activity 类中有一个 onCreate 事件方法，一般在该方法中对 Activity 进行初始化。通过 setContentView 方法可以将 View 放到 Activity 上。绑定后，Activity 会显示 View 上的控件。

一个带界面的 Android 应用程序可以由一个或多个 Activity 组成。至于这些 Activity 如何工作，或者它们之间有什么依赖关系，则完全取决于应用程序的业务逻辑。例如，一种典型的设计方案是使用一个 Activity 作为主 Activity（相当于主窗体，程序启动时会首先显示这个 Activity），在这个 Activity 中通过菜单、按钮等方式显示其他的 Activity。在 Android 自带的程序中有很多都是这种类型的。

每一个 Activity 都会有一个窗口，在默认情况下，这个窗口是充满整个屏幕的，也可以将窗口变得比手机屏幕小，或者悬浮在其他窗口上面。Activity 窗口中的可视化组件由 View 及其子类组成，这些组件按照 XML 布局文件中指定的位置在窗口上进行摆放。

1.5.2　Service（服务）

服务没有可视化 UI，但可以在后台运行。例如，当用户进行其他操作时，可以利用服

务在后台播放音乐，或者当来电时，可以利用服务同时进行其他操作。服务类必须从androidapp.Service 继承。

现在举一个非常简单的使用服务的例子。在手机中会经常使用播放音乐的软件，在这类软件中往往会有循环播放或随机播放的功能。虽然在软件中可能会有相应的功能（通过按钮或菜单进行控制），但用户可能会一边放音乐，一边在手机上做其他的事，例如，与朋友聊天、看小说等。在这种情况下，用户不可能当一首音乐放完后再回到软件界面去进行重放操作。因此，可以在播放音乐的软件中启动一个服务，由这个服务来控制音乐的循环播放，而且服务对用户是完全透明的，这样用户完全感觉不到后台服务的运行，甚至可以在音乐播放软件关闭的情况下，仍然可以播放后台背景音乐。

此外，其他程序还可以与服务进行通信。当与服务连接成功后，就可以利用服务中共享出来的接口与服务进行通信了。例如，控制音乐播放的服务允许用户暂停、重放、停止音乐的播放。

1.5.3 Broadcast Receiver（广播接收器）

广播接收器组件的唯一功能是接收广播动作，以及对广播动作做出响应。有很多时候，广播动作是由系统发出的，例如，时区的变化、电池的电量不足、收到短信等。此外，应用程序还可以发送广播动作，例如，通知其他程序数据已经下载完毕，并且这些数据已经可以使用了。

一个应用程序可以有多个广播接收器，所有的广播接收类都需要继承android.content.BroadcastReceiver 类。

广播接收器与服务一样，都没有用户接口，但在广播接收器中可以启动一个 Activity 来响应广播动作，例如，通过显示一个 Activity 对用户进行提醒。当然，也可以采用其他方法或几种方法的组合来提醒用户，如闪屏、震动、响铃、播放音乐等。

1.5.4 Content Provider（内容提供者）

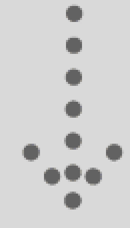
内容提供者可以为其他应用程序提供数据，这些数据可以保存在文件系统中，例如，SQLite 数据库或任何其他格式的文件。每一个内容提供者是一个类，这些类都需要从android.content.ContentProvider 类继承。

在 ContentProvider 类中定义了一系列的方法，通过这些方法可以使其他应用程序获得内容提供者所提供的数据。但在应用程序中不能直接调用这些方法，而需要通过android.Content.ContentResolver 类的方法来调用内容提供者类中提供的方法。

在 Android 系统中很多内嵌的应用程序，如联系人、短信等，都提供了 ContentProvider。其他的应用程序通过这些 ContentProvider 可以对系统内部的数据实现增、删、改操作。例如，可以将指定电话号的短信内容从系统数据库中删除，并将该短信内容加密保存在自己的数据库中，这些删除系统短信的操作就需要通过 Content Provider 来完成。

1.6 本章小结

本章主要介绍了 Android 的发展和特点，介绍了在 Windows 环境搭建 Android 的开发平台，并创建了第一个 Android 应用程序，介绍了 Android 应用程序的框架、资源及应用程序组件。虽然 Android 工程的目录结构较复杂，但并不需要用户手工去建立。如果使用 ADT 来建立工程，会自动生成一个默认的目录结构。当然，可以根据需要对这个默认生成的目录结构进行修改。在 Android 应用程序中包含了大量的资源，这些资源主要保存在 res 目录的子目录中，在生成 apk 文件时会将这些资源一起打包到 apk 文件中。介绍了 Android 应用程序的 4 大应用程序组件：Activity、Service、Broadcast Receiver 和 Content Provider，通过这 4 种应用程序组件可以实现所有类型的 Android 应用程序。这一部分主要是 Android 程序开发的基础，希望读者能打好基础，在接下来的学习中加以体会。



第 2 章　Android 界面布局及基本控件

随着 Android 应用程序的使用越来越广泛，如何设计一款华丽、优雅、体面的应用程序界面变得越来越重要。因此用户需要了解 UI 设计，本章主要学习 Android 屏幕布局及常用基本控件的使用。

2.1　视图 View 概述

View 是 Android 中最基本的一个类，布局（Layout）和控件（Widget）都是继承自 View 类。View 对象是 Android 平台上表示用户界面的基本单元。视图（View）是一个矩形区域，它负责该区域中的绘制和事件处理。视图组（ViewGroup）是视图的子类，是一个容器，专门负责布局。视图组本身没有可绘制的元素。

2.2　Android 界面布局

View 一般分为以下几种布局：线性布局（LinearLayout）、相对布局（RelativeLayout）、表格布局（TableLayout）、单帧布局（FrameLayout）和绝对布局（AbsoluteLayout）。本节主要学习 LinearLayout、RelativeLayout 和 TableLayout，布局主要是在 res/layout/activity_main.xml 中编写。

2.2.1　线性布局（LinearLayout）

线性布局是指在该容器内子控件的摆放方式，有垂直布局和水平布局两种方式。一般通过两个属性 android:orientation 和 android:layout_weight 来设置，其属性的作用如表 2-1 所示。

表 2-1　LinearLayout 中两个重要的属性

Xml 属性	描　述
android:orientation	设置布局的线性方向 Horizontal：水平方向，从左到右 Vertical：垂直方向，从上到下
android:layout_weight	设置控件占屏幕的比例，在垂直布局时，代表行距；水平布局时代表列宽；weight 值越大就表示所占比例越大

下面通过简单的案例学习线性布局及属性。

案例：将 Activity 界面分成上、下两部分，上部分占 1/3，下部分占 2/3，然后上部分是用横向的（水平）布局，里面有 3 个 Button ，下部分则是用纵向的（垂直）布局，也放有 3 个 Button。

效果如图 2-1 所示。

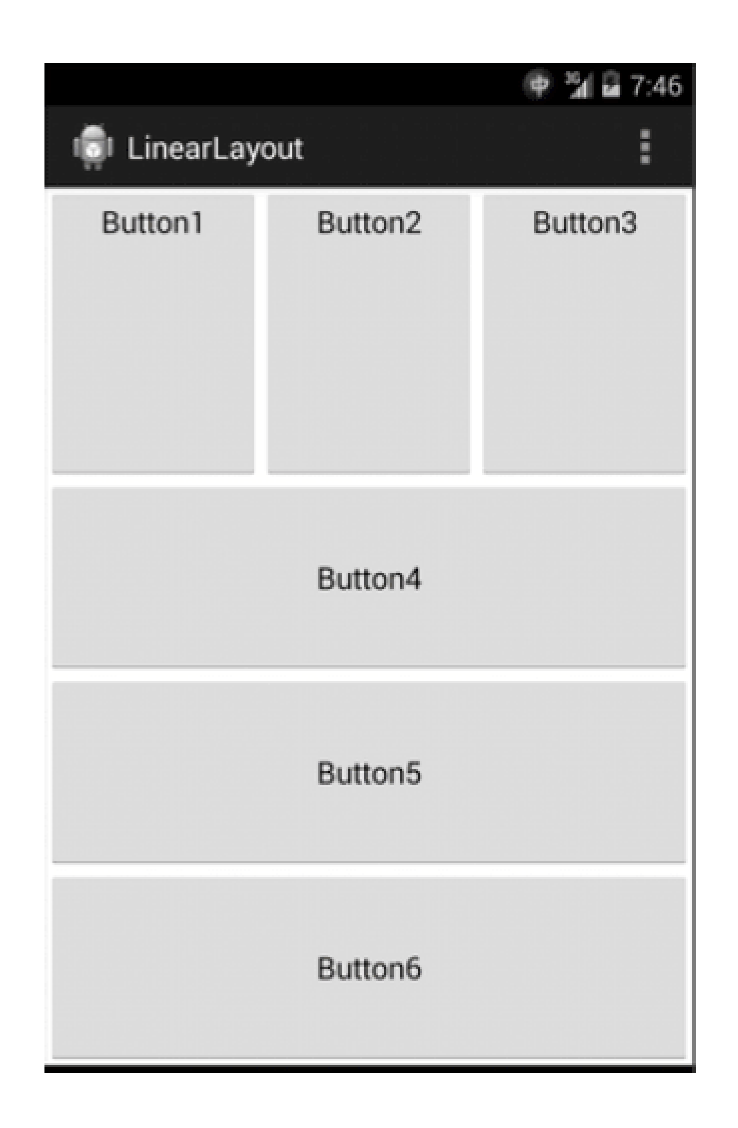


图 2-1　线性布局效果图

案例分析：要实现这样的布局必须要使用到嵌套布局。

首先，最外层是一个垂直布局的 LinearLayout；

其次，在最外层的 LinearLayout 中再嵌套两个（上、下）LinearLayout；

再次，上部分的 LinearLayout 使用水平布局，里面放 3 个 Button；

最后，下部分的 LinearLayout 使用垂直布局，里面放 3 个 Button。

实现步骤如下。

（1）创建一个 Android 工程，工程名为“ch02_Layout”。

（2）在打开“Package Explorer”窗口中的“ch02_Layout”项目中，打开 res/layout/activity_main.xml 文件，修改代码并输入一些代码，代码清单如下。

代码清单：res/layout/activity_main.xml

```
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout width="fill parent"
    android:layout height="fill parent"
     android:orientation="vertical"
   >

 <LinearLayout
      android:orientation="horizontal"
      android:layout width="fill parent"
     android:layout height="fill parent"
     android:layout_weight="2">
```




```
        <Button
          android:gravity="center horizontal"
           android:id="@+id/button1"
            android:layout width="wrap content"
          android:layout height="fill parent"
           android:layout_weight="1"
           android:text="Button1" />

        <Button
           android:gravity="center horizontal"
           android:id="@+id/button2"
          android:layout width="wrap content"
            android:layout height="fill parent"
           android:layout weight="1"
           android:text="Button2" />

        <Button
           android:gravity="center horizontal"
           android:id="@+id/button3"
           android:layout width="wrap content"
         android:layout height="fill parent"
           android:layout weight="1"
           android:text="Button3" />
</LinearLayout>

<LinearLayout
     android:orientation="vertical"
     android:layout width="fill parent"
    android:layout height="fill parent"
    android:layout weight="1" >
        <Button
           android:id="@+id/button4"
           android:layout width="fill parent"
           android:layout height="wrap content"
           android:layout weight="1"
           android:text="Button4" />
        <Button
           android:id="@+id/button5"
           android:layout width="fill parent"
           android:layout_height="wrap_content"
           android:layout weight="1"
           android:text="Button5" />
        <Button
           android:id="@+id/button6"
           android:layout width="fill parent"
           android:layout height="wrap content"
           android:layout weight="1"
           android:text="Button6" />
```

```
    </LinearLayout>
</LinearLayout>
```

布局属性的说明如下。

- **android:layout_width**　设置控件的宽度。
- **android:layout_height**　设置控件的高度。
- **android:layout_weight**　设置控件占屏幕的比例。
- **android:id**　设置控件的 ID。
- **android:text**　设置控件的文本。
- **android:gravity**　设置控件的基本位置。
- **android:background**　设置控件的背景。
- **android:textSize**　设置控件文字的大小。

2.2.2　相对布局（RelativeLayout）

RelativeLayout 这个容器内的子元素都是通过彼此之间的位置来相互定位，或者与其父控件容器进行相互定位。

RelativeLayout 有一些重要的属性，一般分为四组。

第一组：设置控件与给定控件之间的关系和位置，其属性如表 2-2 所示。

表 2-2　控件与给定控件之间的关系和位置的属性

属　　性	描　　述	备　　注
android:layout_above	将该控件置于给定 ID 的控件之上	属性值为某个给定的 ID 例如：android:layout_above="@id/XXX"
android:layout_below	将该控件置于给定 ID 的控件之下	
android:layout_toLeftOf	将该控件置于给定 ID 的控件之左	
android:layout_toRightOf	将该控件置于给定 ID 的控件之右	

第二组：设置控件与控件之间对齐的方式，其属性如表 2-3 所示。

表 2-3　控件与控件之间对齐的方式的属性

属　　性	描　　述	备　　注
Android:layout_alignBaseline	该控件的 baseline 和给定 ID 控件的 baseline 对齐	属性值为某个给定的 ID，例如：android:layout_above="@id/XXX"
android:layout_alignBottom	将该控件的底部边缘与给定 ID 控件的底部边缘对齐	
android:layout_alignLeft	将该控件的左边边缘与给定 ID 控件的左边边缘对齐	
android:layout_alignTop	将该控件的顶部边缘与给定 ID 控件的顶部边缘对齐	
android:layout_alignRight	将该控件的右边边缘与给定 ID 控件的右边边缘对齐	

第三组：设置控件与父控件之间对齐的方式，其属性如表 2-4 所示。

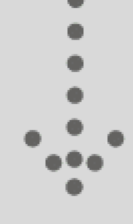


表 2-4　控件与父控件之间对齐的方式的属性

属　　性	描　　述	备　　注
android:layout_alignParentBottom	该控件的底部与父控件的底部对齐	属性值为 true 或 false，这里假设值为 true。如果不指定，默认是 false，表示控件自身的上下左右边缘与父控件的对应边缘是否对齐。由于控件是在父控件的内部，所以是内对齐
android:layout_alignParentLeft	该控件的左边缘与父控件的左边缘对齐	
android:layout_alignParentRight	该控件的右边缘与父控件的右边缘对齐	
android:layout_alignParentTop	该控件的顶部与父控件的顶部对齐	

第四组：设置控件的方向，其属性如表 2-5 所示。

表 2-5　控件的方向的属性

属　　性	描　　述	备　　注
android:layout_centerHorizontal	该控件在其父控件范围内水平居中	属性值为 true 或 false，这里假设值为 true。如果不指定，默认是 false，表示的都是控件自身相对于父控件范围内的居中情况
android:layout_centerInparent	该控件在其父控件范围内垂直且水平居中	
android:layout_centerVertical	该控件在其父控件范围内垂直居中	

下面通过简单的案例来学习相对布局及设置控件的属性。

案例：运用 RelativeLayout（相对布局）制作一副由 9 副小图片组合的大图片，其效果如图 2-2 所示。

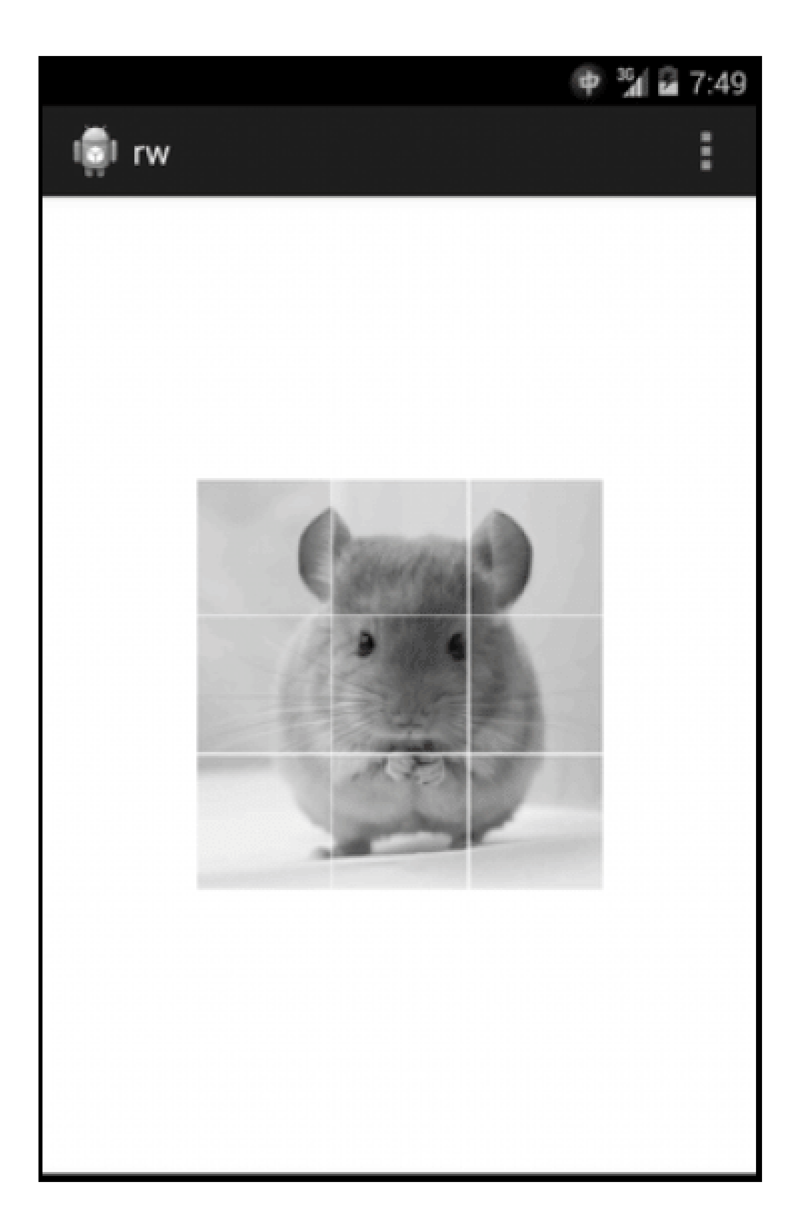


图 2-2　相对布局效果图



用 9 个 TextView，在 TextView 中要显示的图片的属性是 android:background="@drawable/图片名"。

案例分析：要实现这样的布局采用相对布局。



首先，导入项目中所要用到的图片。

其次，打开 activity_main.xml 文件进行相对布局。

最后，在布局中放 9 个 TextView 控件，第一个控件放在界面中间，其他控件以第一个控件为参照物，放到相对应的位置。

实现步骤如下。

（1）创建一个 Android 工程，工程名为“ch02_Rlayout”。

（2）导入 9 张图片放到 res/drawable 对应的 5 个文件中。

（3）在打开“Package Explorer”窗口中的“ch02_Rlayout”项目中，打开 res/layout/activity_main.xml 文件，修改代码并输入一些代码，代码清单如下。

代码清单：res/layout/activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout width="match parent"
   android:layout height="match parent"
   android:paddingBottom="@dimen/activity vertical margin"
   android:paddingLeft="@dimen/activity horizontal margin"
   android:paddingRight="@dimen/activity horizontal margin"
   android:paddingTop="@dimen/activity vertical margin"
   tools:context=".MainActivity" >

    <TextView
      android:id="@+id/center"
      android:layout width="wrap content"
      android:layout height="wrap content"
      android:layout centerInParent="true"
      android:background="@drawable/five"/>

   <TextView
      android:id="@+id/left"
      android:layout width="wrap content"
      android:layout height="wrap content"
      android:layout above="@id/center"
      android:layout_toLeftOf="@id/center"
      android:background="@drawable/one" />
    <TextView
      android:id="@+id/right"
      android:layout width="wrap content"
      android:layout height="wrap content"
      android:layout above="@id/center"
      android:layout toRightOf="@id/center"
      android:background="@drawable/three" />"


   <TextView
      android:id="@+id/unright"
      android:layout width="wrap content"
      android:layout_height="wrap_content"
```

```
        android:layout below="@id/center"
        android:layout_toLeftOf="@id/center"
        android:background="@drawable/serven" />

    <TextView
        android:layout_width="wrap_content"
        android:layout height="wrap content"
        android:layout below="@id/center"
        android:layout toRightOf="@id/center"
        android:background="@drawable/night"/>
     <TextView
        android:layout width="wrap content"
        android:layout height="wrap content"
        android:layout toRightOf="@id/center"
        android:layout below="@id/right"
        android:background="@drawable/six" />"
       <TextView
        android:layout_width="wrap_content"
        android:layout height="wrap content"
        android:layout toLeftOf="@id/center"
        android:layout below="@+id/left"
        android:background="@drawable/four"/>
       <TextView
        android:layout width="wrap content"
        android:layout height="wrap content"
        android:layout above="@id/center"
        android:layout toRightOf="@+id/left"
        android:background="@drawable/two"/>
          <TextView
        android:layout_width="wrap_content"
        android:layout height="wrap content"
        android:layout below="@id/center"
        android:layout toRightOf="@id/unright"
        android:background="@drawable/eight"/>"

</RelativeLayout>
```

2.2.3 表格布局（TableLayout）

TableLayout 中每一行为一个 TableRow 对象，TableRow 可以添加子控件，每添加一个为一列。

TableLayout 中有几个重要的属性，其作用如表 2-6 所示。

表 2-6 TableLayout 的属性

Xml 属性	描 述
android:stretchColumns	指定拉伸列，（从 0 开始计数），当所有列的内容不能填满整个 TableLayout 时，会拉伸指定列，使其宽度变宽，来达到填满整个父控件的目的
anroid:layout_column	控件在 TableRow 中所处的列
android:layout_span	该控件所跨越的列数
android:collapseColumns	将里面指定的列隐藏，若有多列需要隐藏，用逗号将列序号隔开

下面通过简单的案例学习表格布局及设置控件的属性。

案例：用 TableLayout 编写如图 2-3 所示的界面。

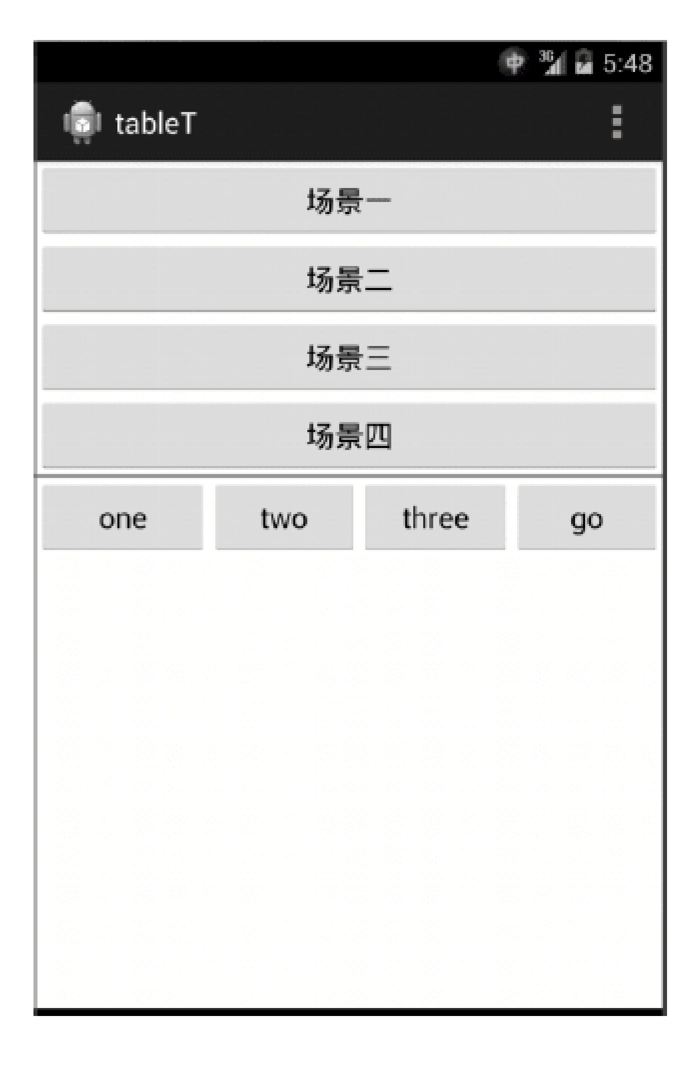


图 2-3　表格布局效果图

案例分析：本案例中采用表格布局来实现。

首先，在界面中所要显示的字符在 res/layout/strings.xml 文件中定义。

然后，在 res/layout/activity_main.xml 文件中布局：添加 5 个 TableRow 对象，前面 4 个 TableRow 对象中分别放入一个 Button 控件，后面一个 TableRow 对象放入 4 个 Button 控件。

实现步骤如下。

（1）创建一个 Android 工程，工程名为“ch02_Tlayout”。

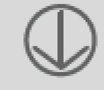
（2）在打开“Package Explorer”窗口中的“ch02_Tlayout”项目中，打开 res/layout/strings.xml 文件，修改代码并输入一些代码，代码清单如下。

代码清单：res/layout/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="app_name">tableT</string>
  <string name="action settings">Settings</string>
  <string name="hello world">Hello world!</string>
   <string name="changjiyi">场景一</string>
  <string name="changjier">场景二</string>
  <string name="changjisan">场景三</string>
  <string name="changjisi">场景四</string>
  <string name="one">one</string>
  <string name="two">two</string>
  <string name="three">three</string>
  <string name="go">go</string>
</resources>
```

说明：在项目中所要显示的文本一般情况下是在 strings.xml 文件中设置的。





（3）打开 res/layout/activity_main.xml 文件，修改代码并输入一些代码，代码清单如下。

代码清单：res/layout/activity_main.xml

```
  <?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
    >
   <TableRow >
      <Button
         android:id="@+id/b1"
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:layout_weight="1"
         android:text="@string/changjiyi" />
   </TableRow>
   <TableRow >
      <Button
         android:id="@+id/b2"
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:layout_weight="1"
        android:text="@string/changjier" />
   </TableRow>
   <TableRow >
      <Button
         android:id="@+id/b3"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_weight="1"
         android:text="@string/changjisan" />
   </TableRow>
   <TableRow >
      <Button
         android:id="@+id/b4"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_weight="1"
         android:text="@string/changjisi" />
   </TableRow>
   <View
      android:layout_height="2dip"
      android:background="#FF909090" />
   <TableRow >
      <Button
         android:id="@+id/b5"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_weight="1"
         android:text="@string/one" />
```

```
        <Button
            android:id="@+id/b6"
            android:layout width="wrap content"
            android:layout height="wrap content"
            android:layout weight="1"
            android:text="@string/two" />
        <Button
            android:id="@+id/b7"
            android:layout width="wrap content"
            android:layout height="wrap content"
            android:layout weight="1"
            android:text="@string/three" />
        <Button
            android:id="@+id/b8"
            android:layout width="wrap content"
            android:layout height="wrap content"
            android:layout_weight="1"
            android:text="@string/go" />
    </TableRow>
</TableLayout>
```

2.3　文本框及按钮控件

本节主要讲 3 个常用控件：TextView（文本显示控件）、EditText（实现文本域，既可以在此文本域中输入内容）和 Button（按钮控件）。有些常用的 android 属性，如表 2-7 所示。

表 2-7　常见的 xml 属性

xml 属性	描　　述
android:layout_width	指定控件的宽度
android:layout_height	指定控件的高度
android:id	为控件指定相应的 Id
android:text	指定控件当中显示的文字，尽量使用 string.xml
android:gravity	指定控件的基本位置
android:textSize	指定控件当中字体的大小
android:background	指定该控件所使用的背景颜色，RGB 命名法，也可以引用 android:drawable 的图片
android:padding	指定控件的内边距（使用 dip 计量最好，因为它不受手机像素屏幕大小的限制，更具有适应性）
android:singleLine	如果设置为 true，则控件当中的内容在同一行显示，多余内容省略号表示

下面通过简单的案例学习 TextView（文本显示控件）、EditText（实现文本域，既可以在此文本域中输入内容）和 Button（按钮控件）。

案例：运用 TextView、EditText 和 Button 控件编写设置邮箱的界面，如图 2-4 所示。



图 2-4　常用控件效果图

案例分析：本案例中采用表格布局来实现。

首先在界面中所要显示的字符在 res/layout/strings.xml 文件中定义。

然后在 res/layout/activity_main.xml 文件中布局：使用相对布局。添加 3 个 TextView 控件、两个 EditText 控件和两个 Button 控件。

实现步骤如下。

（1）创建一个 Android 工程，工程名为“ch02_kongjian”。

（2）在打开“Package Explorer”窗口中的”ch02_kongjian”项目中，打开 res/layout/strings.xml 文件，修改代码并输入一些代码，代码清单如下。

代码清单：res/layout/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">简易邮箱界面</string>
    <string name="action_settings">Settings</string>
    <string name="hello world">Hello world!</string>
    <string name="mail">Mail</string>
    <string name="address">账号</string>
    <string name="pwd">密码</string>
    <string name="submit">确认</string>
    <string name="no">取消</string>
</resources>
```

（3）打开 res/layout/activity_main.xml 文件，修改代码并输入一些代码，代码清单如下。

代码清单：res/layout/activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout width="wrap content"
   android:layout height="wrap content"
   android:orientation="vertical" >
```

```
<TextView
    android:id="@+id/tvt"
    android:layout width="wrap content"
    android:layout height="wrap content"
    android:layout alignParentTop="true"
    android:text="@string/mail"
    android:textSize="80px" >
</TextView>

<Button
    android:id="@+id/btn cacel"
    android:layout width="wrap content"
    android:layout height="wrap content"
    android:layout alignBaseline="@+id/btn login"
    android:layout alignBottom="@+id/btn login"
    android:layout marginLeft="44dp"
    android:layout_toRightOf="@+id/btn_login"
    android:text="@string/no" />

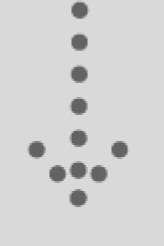
<TextView
    android:id="@+id/tv password"
    android:layout width="wrap content"
    android:layout height="wrap content"
    android:layout below="@+id/tv username"
    android:layout marginTop="48dp"
    android:text="@string/pwd"
    android:textSize="40px" />

<TextView
    android:id="@+id/tv username"
    android:layout width="wrap content"
    android:layout height="wrap content"
    android:layout below="@+id/tvt"
    android:layout marginTop="24dp"
    android:text="@string/address"
    android:textSize="40px" />

<EditText
    android:id="@+id/txt password"
    android:layout width="fill parent"
    android:layout height="wrap content"
    android:layout alignParentLeft="true"
    android:layout below="@+id/tv username"
    android:ems="10" >

    <requestFocus />
</EditText>

<EditText
    android:id="@+id/EditText01"
    android:layout_width="fill_parent"
```



```
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_below="@+id/tv_password"
        android:ems="10" />

    <Button
        android:id="@+id/btn_login"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/EditText01"
        android:layout_marginLeft="30dp"
        android:layout_marginTop="44dp"
        android:layout_toRightOf="@+id/tv_password"
        android:text="@string/submit" />

</RelativeLayout>
```

2.4　应用实例——简单计算器

案例描述：编写一个加减乘除的程序，效果如图 2-5～图 2-7 所示。



图 2-5　简单计算器的初始界面



图 2-6　输入数据

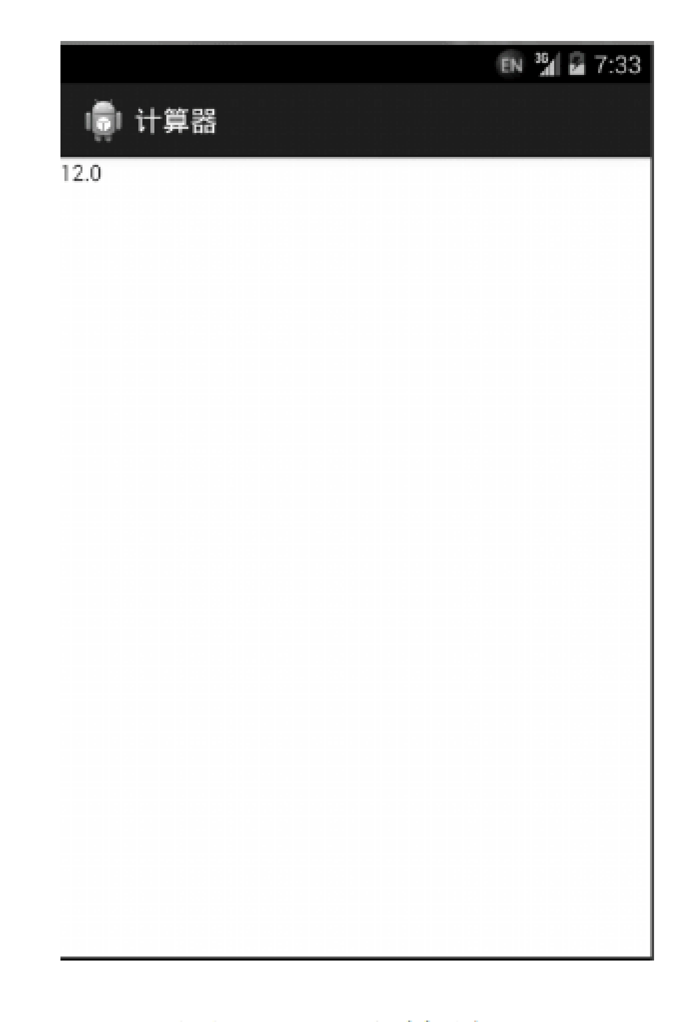
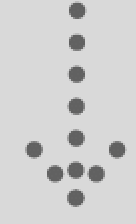


图 2-7　计算结果



案例分析：实现计算器案例要用到两个界面，计算界面和计算结果界面，则必须要有两个 Java 文件（MainActivity.java 文件和 ResultActivity.java 文件），与此相对应的是 activity_main.xml 布局文件和 activity_result.xml 文件。

实现步骤如下。

（1）创建一个 Android 工程，工程名为“ch02_jsq”。

（2）在打开“Package Explorer”窗口中的“ch02_jsq”项目中，创建一个 ResultActivity.java 文件，创建方法为右击 src，在弹出的快捷菜单中选中 New→class 选项，在弹出的对话框中的 Name 后面的文本框中输入 ResultActivity，如图 2-8 所示。



图 2-8　创建 ResultActivity.java 文件的步骤

单击“Finish”按钮，ResultActivity.java 文件创建成功，如图 2-9 所示。

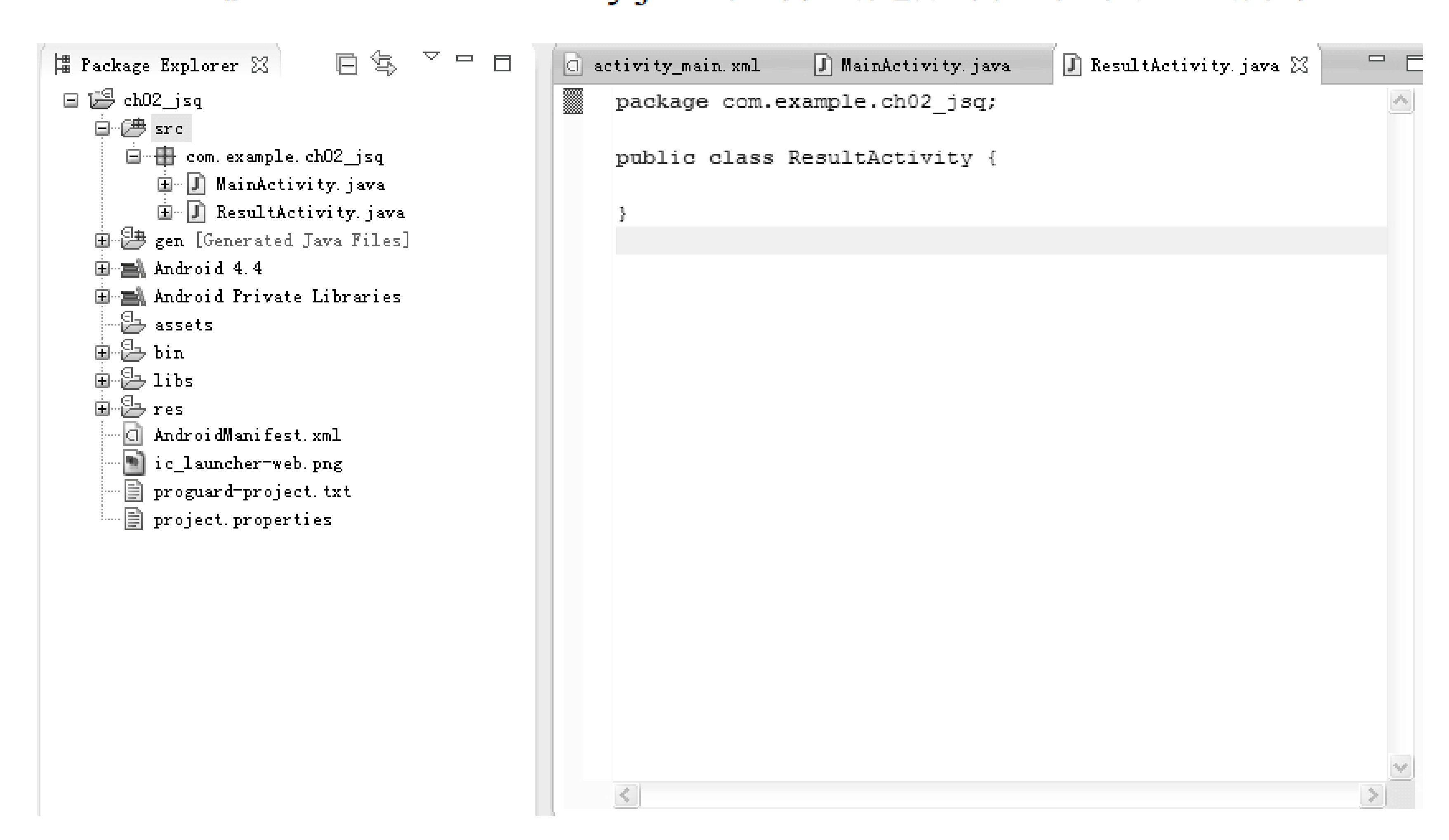


图 2-9　创建 ResultActivity.java 文件

（3）根据图 2-5～图 2-7 所示的界面，打开 res/layout/strings.xml 文件，在里面定义要显示的字符变量，代码清单如下。

代码清单：res/layout/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <string name="app name">计算器</string>
   <string name="action settings">Settings</string>
   <string name="title_activity_main">计算器</string>
   <string name="activity_main_menu1">退出</string>
   <string name="activity_main_menu2">关于</string>
   <string name="activity_main_button11">计算</string>
</resources>
```

（4）根据图 2-5 简单计算器的初始界面布局，打开 res/layout/activity_main.xml 文件，修改代码并输入一些代码，代码清单如下。

代码清单：res/layout/activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout width="fill parent"
   android:layout height="fill parent"
   android:orientation="vertical" >

   <EditText
      android:id="@+id/et1"
      android:layout width="fill parent"
      android:layout height="wrap content"
      android:numeric="integer" />

   <Spinner
      android:id="@+id/sp1 "
      android:layout width="fill parent"
      android:layout height="wrap content" />

   <EditText
      android:id="@+id/et2"
      android:layout width="fill parent"
      android:layout height="wrap content"
      android:numeric="integer" />

   <Button
      android:id="@+id/bt1"
      android:layout width="fill parent"
      android:layout height="wrap content"
      android:text="@string/button01" />

</LinearLayout>
```

（5）创建一个对应 ResultActivity.java 的布局文件 result_main.xml 文件，根据图 2-7 的计算结果布局。创建 result_main.xml 方法为右击 layout，在弹出的快捷菜单中选中 New→file 选项，在弹出的对话框的 File name 后面的文本框中输入 result_main.xml，如图 2-10 所示。



图 2-10　创建 result_main.xml 文件的步骤

单击“Finish”按钮，result_main.xml 文件创建成功，在此文件中输入代码，代码清单如下。

代码清单：res/layout/result_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout width="fill parent"
    android:layout height="fill parent"
    android:orientation="vertical" >
    <TextView
        android:id="@+id/text1"
        android:layout width="fill parent"
        android:layout height="wrap content" />
</LinearLayout>
```

（6）打开 src/com.example.js_jsq/MainActivity.java 文件，修改代码并输入一些代码，代码清单如下。

代码清单：src/com.example.ch02_jsq/MainActivity.java

```
package com.example.ch02 jsq;
import android.os.Bundle;
```




```
import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.view.KeyEvent;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.Toast;

public class MainActivity extends Activity {
    EditText editText1;
    EditText editText2;
    Button button1;
    Spinner spinner1;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        editText1 = (EditText) findViewById(R.id.edittext1);
        editText2 = (EditText) findViewById(R.id.edittext2);
        button1 = (Button) findViewById(R.id.button1);
        spinner1 = (Spinner) findViewById(R.id.spinner1);
        String name[] = { "*", "/", "+", "-" };
        ArrayAdapter adapter1 = new ArrayAdapter(this,
        android.R.layout.simple_spinner_item, name);
        spinner1.setAdapter(adapter1);
        button1.setOnClickListener(new button1_click());

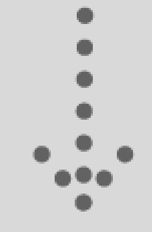
    }
    class button1_click implements OnClickListener {
        public void onClick(View v) {
            String str1 = editText1.getText().toString().trim();
            String str2 = editText2.getText().toString().trim();
            String str3 = spinner1.getSelectedItem().toString().trim();
            if (str1.equals("") || str2.equals("")) {
                Toast toast1 = Toast.makeText(getApplicationContext(),
                        "数值不能为空！", Toast.LENGTH_SHORT);
                toast1.show();
            } else if (str3.equals("/")
                    && (str2.equals("") || str2.equals("0"))) {
                Toast toast1 = Toast.makeText(getApplicationContext(),
                        "被除数不能为空或者为零！", Toast.LENGTH_SHORT);
                toast1.show();
            } else {
                float f1 = Float.valueOf(str1);
```

```
                float f2 = Float.valueOf(str2);
                float f3 = 0;
                if (str3.equals("*")) {
                    f3 = f1 * f2;
                } else if (str3.equals("/")) {
                    f3 = f1 / f2;
                } else if (str3.equals("+")) {
                    f3 = f1 + f2;
                } else {
                    f3 = f1 - f2;
                }
                String string1 = f3 + "";
                Intent intent = new Intent();
                intent.putExtra("str", string1);
                intent.setClass(MainActivity.this, FullActivity.class);
                MainActivity.this.startActivityForResult(intent, 0);
            }
        }
    }
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == R.id.action settings) {
            onKeyDown(KeyEvent.KEYCODE_BACK, null);// 调用 Home 键
        } else if (item.getItemId() == R.id.action_settings) {
            AlertDialog isExit = new AlertDialog.Builder(this).create();
            isExit.setTitle("关于");
            isExit.setMessage("作者：XXXXX");
            isExit.show();
        }
        return true;
    }

    public boolean onKeyDown(int keyCode, KeyEvent event) {
        if (keyCode == KeyEvent.KEYCODE BACK) {
            AlertDialog isExit = new AlertDialog.Builder(this).create();
            isExit.setTitle("提示");
            isExit.setMessage("是否退出？");
            //添加选择按钮并注册监听
            isExit.setButton("确定", listener);
            isExit.setButton2("取消", listener);
            isExit.show();
        }
        return false;
    }

    //监听对话框里面的 button 单击事件
    DialogInterface.OnClickListener listener = new DialogInterface.
```



```
        OnClickListener() {
            public void onClick(DialogInterface dialog, int which) {
                if (which == AlertDialog.BUTTON_POSITIVE) {
                    finish();
                }// AlertDialog.BUTTON_NEGATIVE:
            }
        };
    }
```

（7）打开 src/com.example.js_jsq/ResultActivity.java 文件，修改代码并输入一些代码，代码清单如下。

代码清单：src/com.example.ch02_jsq/ResultActivity.java

```
package com.example.ch02_jsq;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.widget.TextView;

public class ResultActivity extends Activity {
    private TextView text1=null;
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_result);
        text1=(TextView)findViewById(R.id.text1);
        Intent intent=getIntent();
        text1.setText(intent.getStringExtra("str"));
            }
}
```

（8）打开 AndroidManifest.xml 文件，把前面创建的 ResultActivity.java 文件注册。打开 AndroidManifest.xml 文件，输入一些代码，代码清单如下。

代码清单：AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.js_jsq"
   android:versionCode="1"
   android:versionName="1.0" >

   <uses-sdk
      android:minSdkVersion="8"
      android:targetSdkVersion="18" />

   <application
      android:allowBackup="true"
      android:icon="@drawable/ic_launcher"
      android:label="@string/app_name"
      android:theme="@style/AppTheme" >
```

```
        <activity
            android:name="com.example.js jsq.MainActivity"
            android:label="@string/app name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity
            android:name="com.example.js jsq. ResultActivity "
            android:label="@string/app name"

            ></activity>
    </application>

</manifest>
```

（9）程序运行，即可实现一个计算器的功能。

相关知识点：创建 Activity 的要点。以创建案例中的 ResultActivity.java 为例。

① 创建一个名为 ResultActivity 的类（这个类相当于是一个 Activity），并且这个类要继承 Activity。

```
public class ResultActivity extends Activity  {   }
```

② 需要重写 onCreat()，创建 onCreate 方法的步骤为右击 ResultActivity.java，在弹出的快捷菜单中选中 Source→Overrid/Implement Methods…选项，在弹出的对话框中选中 onCreate(Bundle savedInstanceState)，如图 2-11、图 2-12 所示。

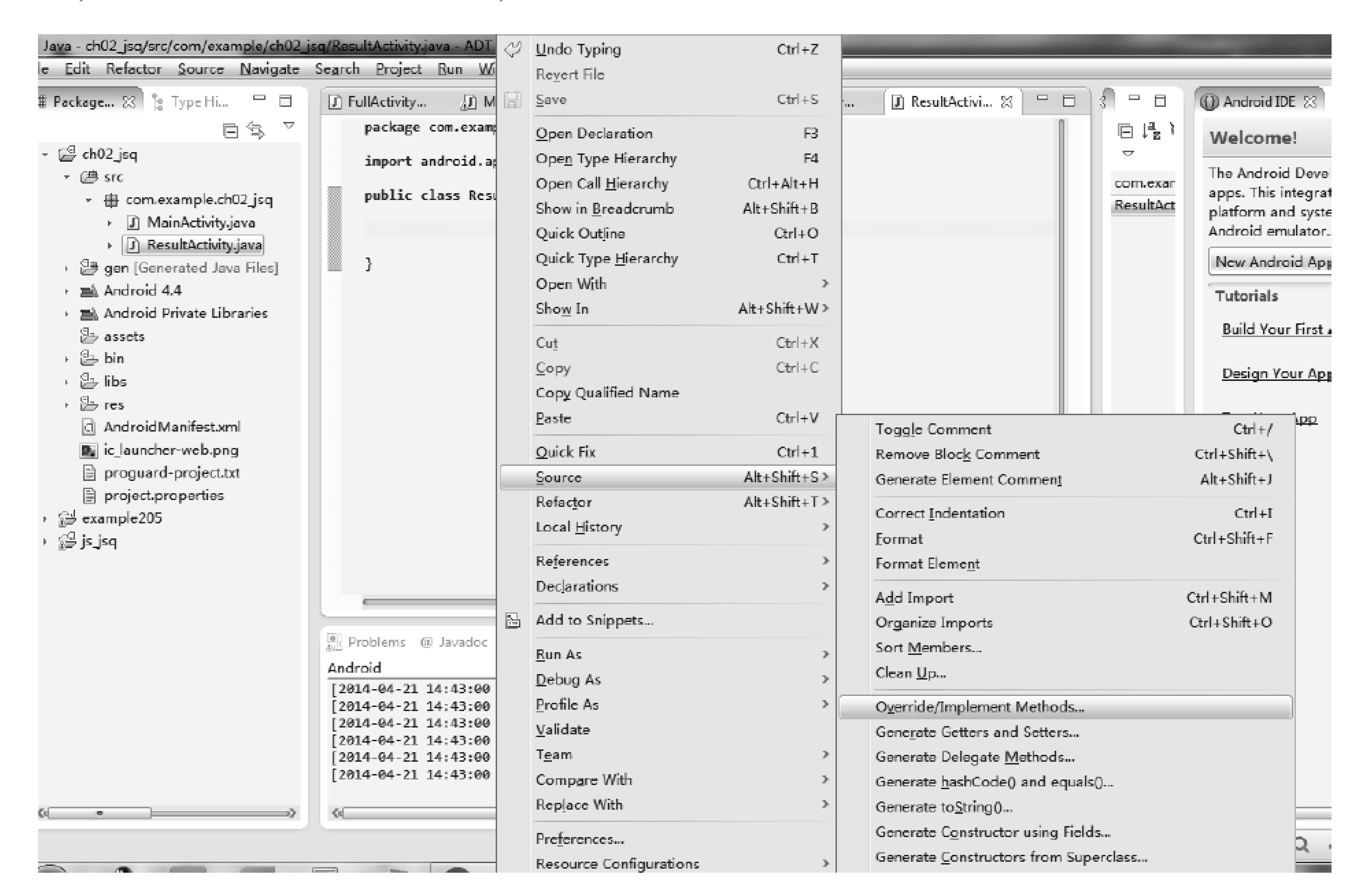


图 2-11　选择命令

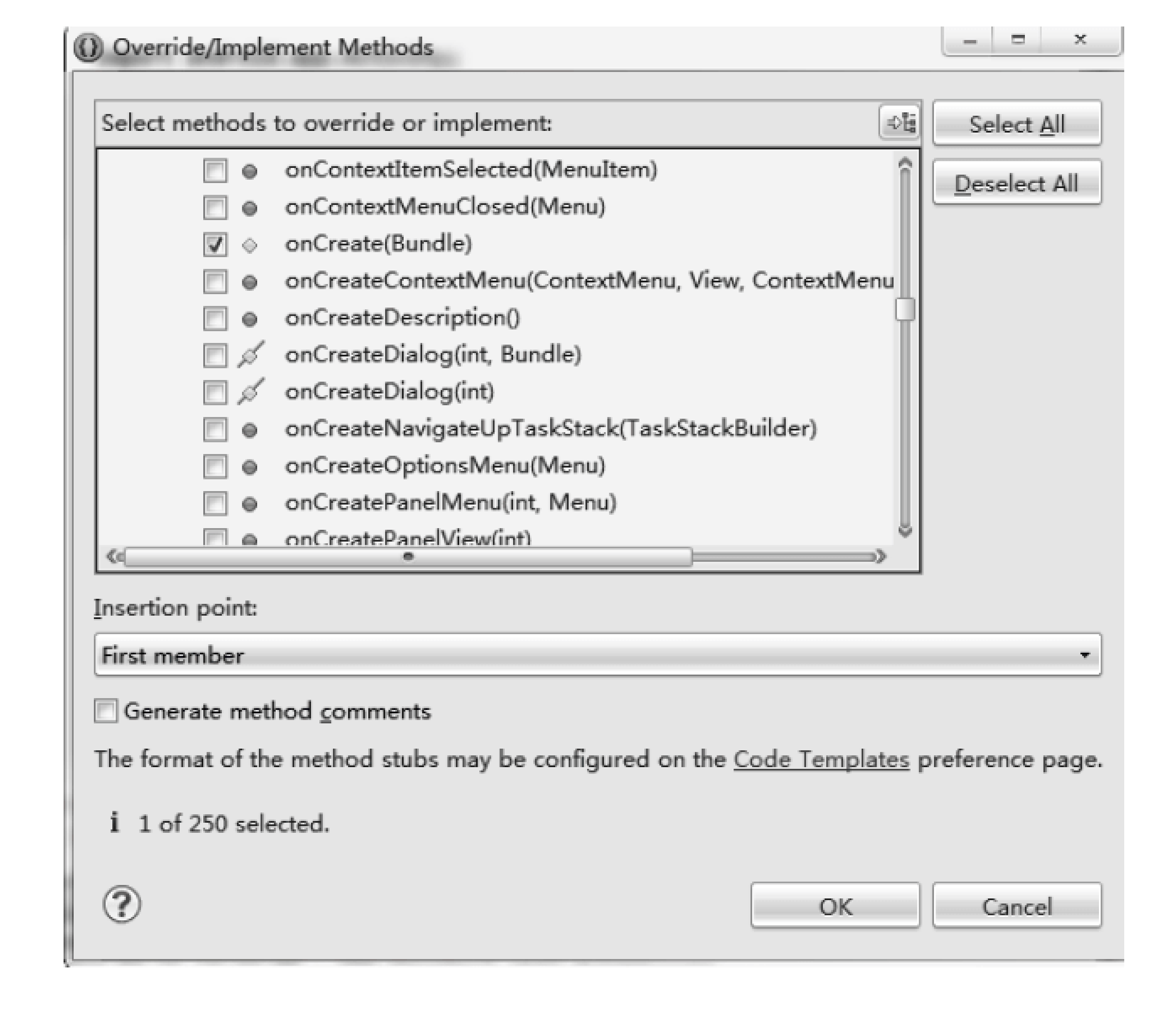


图 2-12　选择 onCreate 选项

③ 每一个 Activity 都需要在 AndroidManifest.xml 文件当中进行配置，参照步骤（8）。

④ 每一个 Activity 应对应一个布局文件（在 res→layout 里创建布局文件），在布局文件中添加必要的控件，参照步骤（5）（多个 Activity 可以共用同一个布局文件）。

2.5　本章小结

本章主要讲解了 3 个布局：线性布局、相对布局和表格布局，以及各布局中所要用到的属性。介绍了 Textview、Editview 和 Button 控件的使用，介绍了控件的一些常用属性。最后通过一个简单计算器的实现，讲解界面之间的跳转、Activity 的创建及按钮的监听类的实现。本章主要以案例的形式来讲解，易于初学者理解，请在开发工具中多调试本章的案例。

第 3 章　Android 控件进阶

要设计出让用户喜欢的 Android 应用程序界面，除了需要用到在第 2 章讲的最基本的 TextView、EditText 和 Button 控件外，还要用到其他控件，如 ImageButton 控件、ImageView 控件、RadioButton 控件、CheckBox 控件和 ListView 控件等。本章主要讲解功能强大、应用广泛的一些控件。

3.1　ImageButton 控件

Android 系统自带的除了在第 2 章中 Button 按钮外，还提供了带图表标的按钮 ImageButton。制作带图标的按钮，首先要在布局文件中定义 ImageButton，然后通过以下几种方法设置要显示的图标。

方法一：在布局文件中就直接设置按钮的图标，如

```
android:src="@drawable/图片地址及图片名"
```

方法二：使用系统自带的图标，如

```
ImageButton1.ImageDrawable(getResources().getDrawable(R.drawable.icon1);
```

设置完按钮的图标，然后为按钮设置监听类 setOnClickListener。下面通过简单案例学习 ImageButton 控件以及其属性。

案例：使用 ImageButton 按钮设计一个界面，效果如图 3-1 所示。



图 3-1　ImageButton 案例效果图

案例分析：

首先，在 activity_main.xml 布局分件布局，添加一个 TextView 控件和 ImageButton 控件，并设置一些属性。

然后，在 MainActivity.java 文件中定义一个变量，通过 findViewById 得到 ImageButton 控件，并添件对应的监听事件。

实现步骤如下。

（1）创建一个 Android 工程，工程名为“ch03_buttonimages”。

（2）在打开“Package Explorer”窗口中的“ch03_buttonimages”项目中，打开 res/layout/activity_main.xml 文件，修改代码并输入一些代码，代码清单如下。

代码清单：res/layout/activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical" android:layout width="fill parent"
    android:layout height="fill parent"
    tools:context=".MainActivity" >
    <TextView
  android:layout width="fill parent"
  android:layout height="wrap content"
  android:id="@+id/textView" />

<ImageButton
android:id="@+id/imageButton"
android:layout width="wrap content"
android:layout height="wrap content">
</ImageButton>
</LinearLayout>
```

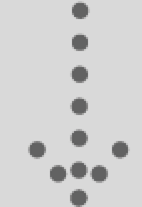
（3）打开 src/com.example.buttonimages/MainActivity.java 文件，修改代码并输入一些代码，代码清单如下。

代码清单：src/com.example.buttonimages/MainActivity.java

```
package com.example.buttonimages;
import android.os.Bundle;
import android.app.Activity;
import android.view.View;
import android.widget.Button;
import android.widget.ImageButton;
import android.widget.TextView;
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        setTitle("ImageButton");

        ImageButton imgButton = (ImageButton) this.findViewById(R.id.
        imageButton);
        //设置图片按钮的背景
        imgButton.setBackgroundResource(R.drawable.buttonimage);
        //setOnClickListener() - 响应图片按钮的鼠标单击事件
        imgButton.setOnClickListener(new Button.OnClickListener(){
        @Override
```

```
        public void onClick(View v) {
        TextView txt = (TextView)  MainActivity.this.findViewById
        (R.id.textView);
        txt.setText(R.id.txtview);
        }
        });
    }}
```



3.2 ImageView 控件

ImageView 控件是 Android 中的基础图片显示控件，这也是布局中使用图片最常用的方式，可以使程序变得生动活泼，该控件有个重要的属性是 ScaleType，该属性用以表示显示图片的方式，共有 8 种取值，如表 3-1 所示。

表 3-1 ScaleType 的值

ScaleType 的值	描　述
ScaleType.CENTER	图片大小为原始大小，如果图片大小大于 ImageView 控件，则截取图片中间部分，若小于，则直接将图片居中显示
ScaleType.CENTER_CROP	将图片等比例缩放，让图像的短边与 ImageView 的边长度相同，即不能留有空白，缩放后截取中间部分进行显示
ScaleType.CENTER_INSIDE	将图片大小大于 ImageView 的图片进行等比例缩小，直到整幅图能够居中显示在 ImageView 中，小于 ImageView 的图片不变，直接居中显示
ScaleType.FIT_CENTER	ImageView 的默认状态，大图等比例缩小，使整幅图能够居中显示在 ImageView 中，小图等比例放大，同样要整体居中显示在 ImageView 中
ScaleType.FIT_END	缩放方式同 FIT_CENTER，只是将图片显示在右方或下方，而不是居中
ScaleType.FIT_START	缩放方式同 FIT_CENTER，只是将图片显示在左方或上方，而不是居中
ScaleType.FIT_XY	将图片非等比例缩放到大小与 ImageView 相同
ScaleType.MATRIX	是根据一个 3×3 的矩阵对其中图片进行缩放

下面通过简单的案例学习 ImageView 控件及其属性。

案例：使用 ImageView 设计一个界面，效果如图 3-2 所示。



图 3-2　ImageView 案例效果图

实现步骤如下。

（1）把图片导入到资源中：将图片拖曳到项目 res\drawable 开头的 5 个文件夹下，它们分别代表了高、中、低分辨度的图片。Android 读取图片时自动优化，选用合适的一个图片显示，比如，高分辨率可以存放 128*128 的图片，低分辨率可以存放 32*32 的图片。

（2）在 string.xml 文件中输入需要显示的字符，打开 res/layout/strings.xml 文件，修改并输入一些代码，代码清单如下。

代码清单：res/layout/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app name">imageV</string>
    <string name="action settings">Settings</string>
     <string name="hello world">hello! </string>
</resources>
```

（3）在 XML 布局文件中添加 ImageView 控件，打开 res/layout/activity_main.xml 文件，修改并输入一些代码，代码清单如下。

代码清单：res/layout/activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout width="match parent"
    android:layout height="match parent"
    android:paddingBottom="@dimen/activity vertical margin"
    android:paddingLeft="@dimen/activity horizontal margin"
    android:paddingRight="@dimen/activity horizontal margin"
    android:paddingTop="@dimen/activity vertical margin"
    tools:context=".MainActivity" >

    <TextView
        android:layout width="wrap content"
        android:layout height="wrap content"
        android:text="@string/hello world"
        android:textSize="80px" />
   <ImageView
       android:src="@drawable/one"
     android:layout width="wrap content"
     android:layout height="wrap content"></ImageView>
</RelativeLayout>
```

3.3 单选按钮与复选框

单选框（RadioButton）、复选框（CheckBox）继承了 Button 类，因此可以直接使用 Button 支持的各种属性和方法。RadioButton、CheckBox 与普通按钮不同之处是多了一个可选中

的功能，因此有个额外的属性，android:checked 属性，该属性用于指定它们初始时是否被选中。

3.3.1 RadioGroup、RadioButton 的用法

RadioGroup 是 RadioButton 的组。每一组 RadioGroup 里至少包含两个 RadioButton，包含多个单选按钮，但只能有一个 RadioButton 被选中，不同的组之间互不影响；每一组 RadioGroup 中都有一个默认的被选中的单选按钮，大部分情况下建议选择第一个为默认选择。

案例：使用 RadioButton 和 RadioGroup 设计一个界面，当选中某个单选框时，弹出相关的一段话，例如，当选中“海陆大餐（好吃真好吃）”，弹出“「山珍海味」，乐不思蜀的人，为人海派，从不拖泥带水，拥有坚忍不拔的性格。但是不够冷静、过度挥霍的结果，只怕会坐吃山空，不得不多加警惕”等语句，效果如图 3-3 所示。

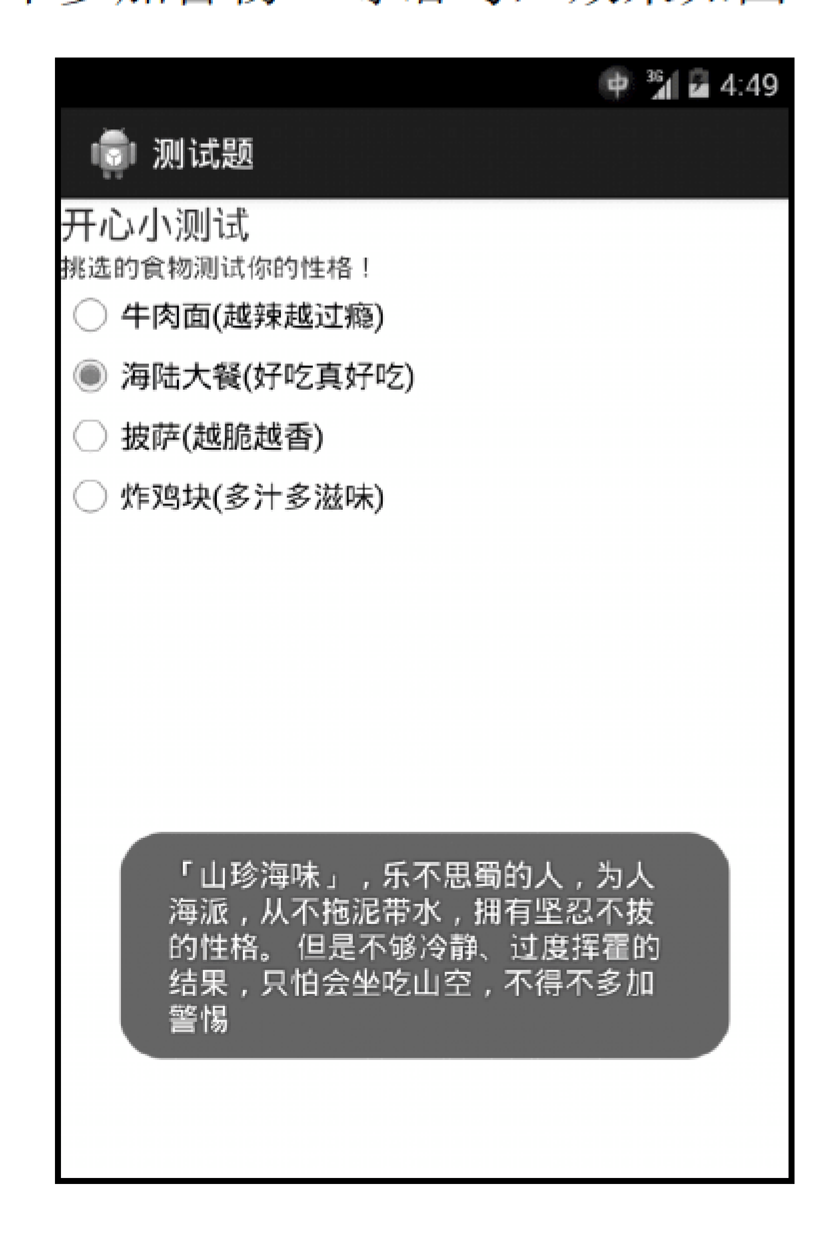


图 3-3　单选按钮效果图

实现步骤如下。

（1）新建一个 Android 应用程序。

（2）编写 string.xml 文件，添加需要显示的字符，打开 res/layout/strings.xml 文件，修改并添加一些代码，代码清单如下。

代码清单：res/layout/strings.xml

```
 <?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">测试题</string>
    <string name="action settings">Settings</string>
    <string name="title">开心小测试</string>
```




```
    <string name="choose">挑选的食物测试你的性格! </string>
    <string name="niu">牛肉面(越辣越过瘾)</string>
    <string name="hai">海陆大餐(好吃真好吃)</string>
    <string name="pizza">披萨(越脆越香)</string>
    <string name="zha">炸鸡块(多汁多滋味) </string></resources>
```

（3）编写 activity_main.xml 文件，添加一个 RadioGroup 标，在 RadioGroup 标签内添加四个 RadioButton ，打开 res/layout/activity_main.xml 文件，修改并添加一些代码，代码清单如下。

代码清单：res/layout/activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout width="match parent"
    android:layout height="match parent"
    android:orientation="vertical"
    >
    <TextView
        android:layout width="wrap content"
        android:layout height="wrap content"
        android:text="@string/title"
        android:textSize="40px" />

   <TextView
        android:id="@+id/who"
        android:layout width="wrap content"
        android:layout height="wrap content"
        android:text="@string/choose"

         />
    <RadioGroup
        android:id="@+id/ceshi group"
          android:layout width="wrap content"
          android:layout_height="wrap_content"
          android:orientation="vertical"
      >
       <RadioButton
           android:id="@+id/niunan"
           android:layout height="wrap content"
           android:layout width="wrap content"
           android:text="@string/niu"
           />
       <RadioButton
           android:id="@+id/hailu"
           android:layout width="wrap content"
           android:layout height="wrap content"
           android:text="@string/hai"
           />
       <RadioButton
           android:id="@+id/pizza"
```



```
            android:layout width="wrap content"
            android:layout height="wrap content"
            android:text="@string/pizza"
            />
        <RadioButton
            android:id="@+id/zhaji"
            android:layout width="wrap content"
            android:layout height="wrap content"
            android:text="@string/zha"
            />
        </RadioGroup>

</LinearLayout>
```

（4）编写 Activity，先声明 6 个全局变量，用于接收这 6 个控件对象，在 onCreate()方法内，根据控件 id 获得这 6 个对象并赋给相应的变量，编写监听器，打开 src/com.example. sumothers /MainActivity.java 文件，修改并添加一些代码，代码清单如下。

代码清单：src/com.example. sumothers /MainActivity.java

```
 package com.example.sumothers;

import android.app.Activity;
import android.os.Bundle;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.CompoundButton.OnCheckedChangeListener;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends Activity {

    //定义各控件的变量
    private TextView who = null;
    private TextView how = null;
    private RadioGroup ceshi group = null;
    private RadioButton niunan = null;
    private RadioButton hailu = null;
    private RadioButton pizza = null;
    private RadioButton zhaji = null;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity main);

        //获得对应的控件
        who = (TextView)findViewById(R.id.who);
        ceshi_group = (RadioGroup)findViewById(R.id.ceshi_group);
```

```
        niunan = (RadioButton)findViewById(R.id.niunan);
        hailu = (RadioButton)findViewById(R.id.hailu);
        pizza = (RadioButton)findViewById(R.id.pizza);
       zhaji = (RadioButton)findViewById(R.id.zhaji);

        //设置 ceshi_group 的监听器，其实是一句代码，其参数是一个带有重构函数的对象
        ceshi group.setOnCheckedChangeListener(new RadioGroup.
        OnCheckedChangeListener() {
              public void onCheckedChanged(RadioGroup group, int checkedId) {
               // TODO Auto-generated method stub
               if(checkedId == niunan.getId()){
                   Toast.makeText(MainActivity.this,"吃辛辣食物的人，本身也很
                   「辣」，性情孤傲，愤世嫉俗，对社交活动、对礼尚往来极端排斥，但对立
                    大功、成大业，成为名流，永垂青史的英雄，欲意气风发、不落人后。东北
                    人多半具有如此的「风格」。 ", Toast.LENGTH_LONG).show();
               }
               else if(checkedId == hailu.getId()){
                   Toast.makeText(MainActivity.this, "「山珍海味」，乐不思蜀的
                   人，为人海派，从不拖泥带水，拥有坚忍不拔的性格。但是不够冷静、过度
                   挥霍的结果，只怕会坐吃山空，不得不多加警惕", Toast.LENGTH
                   LONG).show();
               }
               else if(checkedId == pizza.getId()){
                   Toast.makeText(MainActivity.this, "喜欢吃「薄饼」的人，为人
                   也比较刻薄小气，在团体中属于叛逆的角色，有点自以为是。但是，杰出的
                   艺术家、科学家都具有此种「风格」。", Toast.LENGTH_LONG).show();
               }
               else if(checkedId == zhaji.getId()){
                   Toast.makeText(MainActivity.this, " 這种人属于不爱动的后现
                   代主义者，感情「脆」弱、深怕寂寞，举手投足像只小绵羊一般温驯，欠缺
                   冲劲。 ", Toast.LENGTH LONG).show();
               }
            }
        });
    }
}
```

（5）运行程序，即可得到相应的效果。

知识点：

（1）监听器实现的是 RadioGroup.OnCheckedChangeListener()提供的接口，需要重写里面的 public void onCheckedChanged(RadioGroup group, int checkedId) 方法，该方法的第一个参数用来接收 RadioGroup 对象，第二个参数用来接收被选中的 RadioButton 的 ID。在这个方法中可以做一系列的判断和操作，如判断 RadioButton 的 id 是否等于 checkedId，如果等于就使用 Toast 显示提示消息。

（2）Toast 是 Android 中用来显示显示信息的一种机制，和 Dialog 不同的是，Toast 是没有焦点的，而且 Toast 显示的时间有限，过一定的时间就会自动消失。

Toast 的使用方法如下。

① 创建 Toast 对象。

```
makeText(Context context, CharSequence text, int duration);
```

通过调用这个方法，返回一个 Toast 对象。

第一个参数是上下文对象，通常是用户的应用程序或 Activity 对象——类名.this，第二个参数就是要显示的文本内容，可以格式化文本，第三个参数是持续多长时间来显示消息，有两个常量：LENGTH_SHORT 或者 LENGTH_LONG。

② 调用 show()方法显示。

```
Toast toast = Toast.makeText(RadioTest.this, "female", Toast.LENGTH_SHORT);
toast.show();
```



将监听器绑定到 RadioGroup 上明确两点：

a. 这里绑定监听器的是 RadioGroup 对象而不是 RadioButton 对象。

b. 这里的监听器实现的是 RadioGroup.OnCheckedChangeListener()提供的接口。

3.3.2 CheckBox 的用法

复选框（CheckBox）是一种双状态的按钮，可以选中或不选中，能同时选择多个，每次单击时可以选择是否被选中，在 UI 中默认的是以矩形方式显示。它不同于单选按钮（RadioButton），一个选项就一个 CheckBox，两个选项就两个 CheckBox。对于事件监听它与 RadioButton 的监听是一样的，同样是通过 CompoundButton.OnCheckedChangeListener 来监听的。在 Java 文件中为每一个 CheckBox 都编写一个监听器，该监听器实现的是 CompoundButton.OnCheckedChangeListener()提供的接口，需要重写里面的 public void onCheckedChanged(CompoundButton buttonView, boolean isChecked)方法，该方法的第一个参数用来接收 CompoundButton 对象，第二个参数是用来接收是否被选中，在该方法中可以做一系列的判断和操作，如判断某个 CheckBox 有没有被选中。

案例：使用 RadioButton 和 RadioGroup、CheckBox 设计一个界面，选中单选按钮显示选中的内容，选中多选按钮，也显示选中的内容，效果如图 3-4 所示。

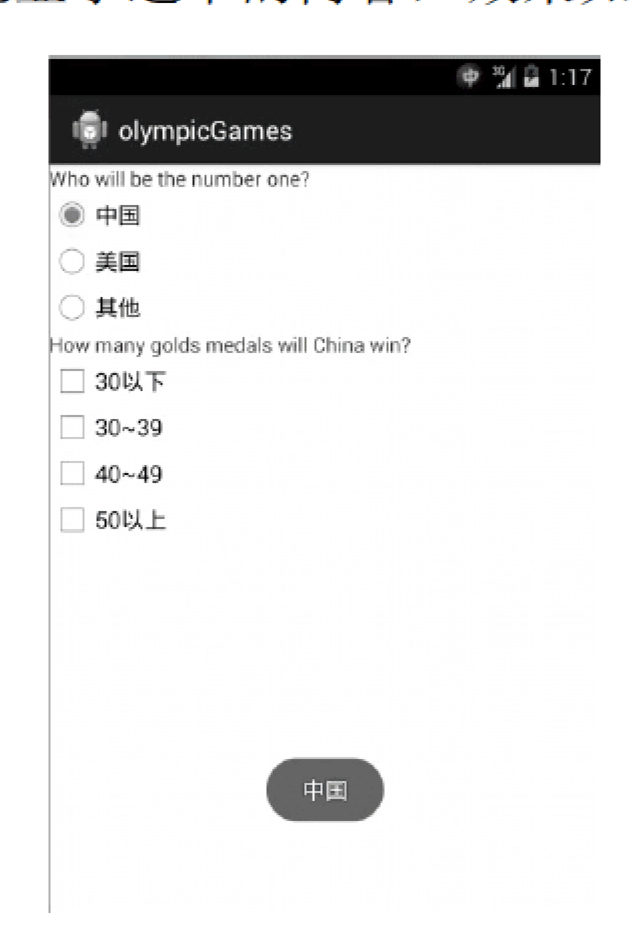


图 3-4　单选、多选按钮效果图



实现步骤如下。

（1）编写 string.xml 文件，添加需要显示的字符，打开 res/layout/strings.xml 文件，修改并添加一些代码，代码清单如下。

代码清单：res/layout/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app name">olympicGames</string>
    <string name="action settings">Settings</string>
    <string name="hello world">Hello world!</string>
    <string name="who">Who will be the number one?</string>
    <string name="china">中国</string>
    <string name="america">美国</string>
    <string name="others">其他</string>
    <string name="how">How many golds medals will China win?</string>
    <string name="less">30 以下</string>
    <string name="thirty">30~39</string>
    <string name="forty">40~49</string>
    <string name="fifty">50 以上</string>
</resources>
```

（2）编写 activity_main.xml 文件，添加一个 RadioGroup 标，在 RadioGroup 标签内添加三个 RadioButton ，添加四个 CheckBox，两个 TextView，修改并添加一些代码，代码清单如下。

代码清单：res/layout/activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout width="match parent"
    android:layout height="match parent"
    android:orientation="vertical"
    >
    <TextView
        android:id="@+id/who"
        android:layout width="wrap content"
        android:layout height="wrap content"
        android:text="@string/who"
         />
    <RadioGroup
        android:id="@+id/who group"
          android:layout width="wrap content"
          android:layout height="wrap content"
          android:orientation="vertical"
      >
       <RadioButton
           android:id="@+id/china"
           android:layout height="wrap content"
           android:layout width="wrap content"
           android:text="@string/china"
```



```
        />
    <RadioButton
        android:id="@+id/america"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/america"
        />
    <RadioButton
        android:id="@+id/others"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/others"
        />
    </RadioGroup>
    <TextView
        android:id="@+id/how"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/how"
        />
     <CheckBox
        android:id="@+id/less"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/less"
        />
    <CheckBox
        android:id="@+id/thirty"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/thirty"
        />
    <CheckBox
        android:id="@+id/forty"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/forty"
        />
    <CheckBox
        android:id="@+id/fifty"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/fifty"
        />

</LinearLayout>
```

（3）编写 Activity，先声明 10 个全局变量，用于接收这 10 个控件对象，在 onCreate()方法内，根据控件 id 获得这 10 个对象并赋给相应的变量，编写监听器，打开 src/ com.example.



olympicgames /MainActivity.java 文件，修改并添加一些代码，代码清单如下。

代码清单：src/com.example.olympicgames/MainActivity.java

```
package com.example.olympicgames;
import android.app.Activity;
import android.os.Bundle;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.CompoundButton.OnCheckedChangeListener;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends Activity {

    //定义各控件的变量
    private TextView who = null;
    private TextView how = null;
    private RadioGroup who group = null;
    private RadioButton china = null;
    private RadioButton america = null;
    private RadioButton others = null;
    private CheckBox less = null;
    private CheckBox thirty = null;
    private CheckBox forty = null;
    private CheckBox fifty = null;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity main);

        //获得对应的控件
        who = (TextView)findViewById(R.id.who);
        how = (TextView)findViewById(R.id.how);
        who group = (RadioGroup)findViewById(R.id.who group);
        china = (RadioButton)findViewById(R.id.china);
        america = (RadioButton)findViewById(R.id.america);
        others = (RadioButton)findViewById(R.id.others);
        less = (CheckBox)findViewById(R.id.less);
        thirty = (CheckBox)findViewById(R.id.thirty);
        forty = (CheckBox)findViewById(R.id.forty);
        fifty = (CheckBox)findViewById(R.id.fifty);

        //设置 who_group 的监听器，其实是一句代码，其参数是一个带有重构函数的对象
        who group.setOnCheckedChangeListener(new RadioGroup.
        OnCheckedChangeListener() {

            public void onCheckedChanged(RadioGroup group, int checkedId) {
                // TODO Auto-generated method stub
```

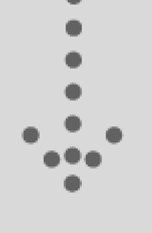
```
            if(checkedId == china.getId()){
                Toast.makeText(MainActivity.this,"中国", Toast.LENGTH_
                SHORT).show();
            }
            else if(checkedId == america.getId()){
                Toast.makeText(MainActivity.this, "美国", Toast.LENGTH_
                SHORT).show();
            }
            else if(checkedId == others.getId()){
                Toast.makeText(MainActivity.this,"其它国家", Toast.LENGTH
                SHORT).show();
            }
        }
    });

    //下面为 4 个 checkbox 多选按钮分别建立监听器
    less.setOnCheckedChangeListener(new OnCheckedChangeListener() {

        public void onCheckedChanged(CompoundButton buttonView, boolean
        isChecked) {
            // TODO Auto-generated method stub
            if(isChecked)
            {
                Toast.makeText(MainActivity.this, "30 个以下", Toast.
                LENGTH SHORT).show();
            }
            else{
                Toast.makeText(MainActivity.this, "不是 30 个以下", Toast.
                LENGTH SHORT).show();
            }
        }
    });


  //下面为 4 个 checkbox 多选按钮分别建立监听器
    thirty.setOnCheckedChangeListener(new CompoundButton.OnChecked
    ChangeListener() {

        public void onCheckedChanged(CompoundButton buttonView, boolean
        isChecked) {
            // TODO Auto-generated method stub
            if(isChecked)
            {
                Toast.makeText(MainActivity.this, "30~39", Toast.LENGTH
                SHORT).show();
            }
            else{
                Toast.makeText(MainActivity.this, "不是 30~39", Toast.
                LENGTH SHORT).show();
            }
        }
```



```
        });

      //下面为 4 个 checkbox 多选按钮分别建立监听器
        forty.setOnCheckedChangeListener(new OnCheckedChangeListener() {

            public void onCheckedChanged(CompoundButton buttonView, boolean
            isChecked) {
                // TODO Auto-generated method stub
                if(isChecked)
                {
                    Toast.makeText(MainActivity.this, "40~49", Toast.LENGTH
                    SHORT).show();
                }
                else{
                    Toast.makeText(MainActivity.this, "不是 40~49", Toast.
                    LENGTH SHORT).show();
                }
            }
        });

      //下面为 4 个 checkbox 多选按钮分别建立监听器
        fifty.setOnCheckedChangeListener(new OnCheckedChangeListener() {

            public void onCheckedChanged(CompoundButton buttonView, boolean
            isChecked) {
                // TODO Auto-generated method stub
                if(isChecked)
                {
                    Toast.makeText(MainActivity.this, "50 以上", Toast.
                    LENGTH SHORT).show();
                }
                else{
                    Toast.makeText(MainActivity.this, "不是 50 以上", Toast.
                    LENGTH SHORT).show();
                }
            }
        });
    }
}
```

3.4 列表视图（ListView）

ListView 是 Android 软件开发中非常重要组件之一，它以列表形式展示具体内容（如联系人），并且能够根据数据的长度自适应显示，每个软件基本上都会使用 ListView.。列表的显示需要如下三个元素。

（1）ListVeiw：用来展示列表的 View。

（2）适配器：用来把数据映射到 ListView 上的中介。一般有三种，ArrayAdapter、

SimpleAdapter 和 SimpleCursorAdapter，其中，以 ArrayAdapter 最为简单，只能展示一行字。SimpleAdapter 有最好的扩充性，可以自定义出各种效果。SimpleCursorAdapter 可以认为是 SimpleAdapter 对数据库的简单结合，可以方便地把数据库的内容以列表的形式展示出来。

（3）数据：指具体的将被映射的字符串、图片、或者基本组件等。

3.4.1　简单的 ListView

在 List 列表中可以直接用 new ArrayAdapter()绘制列表。但如果列表中过于复杂，就需要使用自定义布局来实现 List 列表。

案例：使用 List 列表编写一个界面，当单击某条记录时，用 Toast 显示信息，如图 3-5 所示。

listView
姓名：小王
性别：男
年龄：25
居住地：杭州
邮箱：miswang@gmail.com
联系方式:157571885254
您选择了联系方式:157571885254

图 3-5　简单的 listView 效果图

实现步骤：打开 src/com.example.listview/MainActivity.java 文件， 修改并添加一些代码，代码清单如下。

代码清单：src/com.example.listview/MainActivity.java

```
package com.example.listview;
import android.os.Bundle;
import android.app.ListActivity;
import android.view.Menu;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;

public class MainActivity extends ListActivity {
```




```
        private String[] mListStr = {"姓名：小王","性别：男","年龄：25","居住地：
        杭州","邮箱：miswang@gmail.com","联系方式:157571885254"};
            ListView mListView = null;
            @Override
            protected void onCreate(Bundle savedInstanceState) {
                mListView = getListView();
                setListAdapter(new ArrayAdapter<String>(this,
                        android.R.layout.simple_list_item_1, mListStr));
                mListView.setOnItemClickListener(new OnItemClickListener() {
                    @Override
                   public void onItemClick(AdapterView<?> adapterView, View view,
                   int position,
                           long id) {
                        Toast.makeText(MainActivity.this,"您选择了" + mListStr
                        [position], Toast.LENGTH_SHORT).show();
                    }
                });

                super.onCreate(savedInstanceState);
            }
        }
```

3.4.2 带标题的 ListView 列表

使用 simpleAdapter 时注意要用 Map<String,Object> item 保存列表中每一项显示的 title 与 text，使用 new SimpleAdapter 时将 map 中的数据写入，程序就会自动绘制列表了。

案例：编写一个带标题的 listview 列表，效果如图 3-6 所示。



图 3-6　带标题的 listview 效果图

实现步骤：打开 src/com.example.listviewother/MainActivity.java 文件，修改并添加一些代码，代码清单如下。

代码清单：src/com.example.listviewother/MainActivity.java

```
package com.example.listviewother;

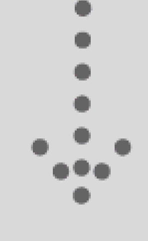
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.os.Bundle;
import android.app.ListActivity;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.SimpleAdapter;
import android.widget.Toast;

public class MainActivity extends ListActivity {

    private String[] mListTitle = { "姓名", "性别", "年龄", "居住地","邮箱",
    "手机号码"};
    private String[] mListStr = { "小胡", "男", "19", "杭州",
           "xiaohu@gmail.com" ,"157571885421"};
    ListView mListView = null;
    ArrayList<Map<String,Object>> mData= new ArrayList<Map<String,Object>>();;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        mListView = getListView();

        int lengh = mListTitle.length;
        for(int i =0; i < lengh; i++) {
            Map<String,Object> item = new HashMap<String,Object>();
            item.put("title", mListTitle[i]);
            item.put("text", mListStr[i]);
            mData.add(item);
        }
        SimpleAdapter adapter = new SimpleAdapter(this,mData,android.R.
        layout.simple_list_item_2,
                new String[]{"title","text"},new int[]{android.R.id.text1,
                android.R.id.text2});
        setListAdapter(adapter);
        mListView.setOnItemClickListener(new OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> adapterView, View view,
            int position,
                    long id) {
                Toast.makeText(MainActivity.this,"您选择了： " + mListTitle
```



```
                  [position] + "内容: "+mListStr[position], Toast.LENGTH LONG).
                  show();
              }
          });
          super.onCreate(savedInstanceState);
      }
  }
```

3.4.3 带图片的 ListView 列表

由于 simpleAdapter 类中的构造函数完成不了带图片的 ListView 列表的界面布局，所以必须自己写布局，使用 Map<String,Object> item 来保存列表中每一项需要的显示内容，如图片、标题、内容等。

案例：编写一个带图片的 listview 列表，效果如图 3-7 所示。



图 3-7 带图片的 listview 的效果图

实现步骤如下。

（1）编写 activity_main.xml 布局文件，添加一个 ImageView 控件，两个 TextView 控件，打开 res/layout/activity_main.xml 文件，修改并添加一些代码，代码清单如下。

代码清单：res/layout/activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
      android:layout width="fill parent" android:layout height=
      "?android:attr/listPreferredItemHeight">
      <ImageView android:id="@+id/tupian"
            android:layout width="wrap content"
            android:layout height="fill parent"
            android:layout alignParentTop="true"
            android:layout_alignParentBottom="true"
```

```
                android:adjustViewBounds="true"
                android:padding="2dip" />
        <TextView android:id="@+id/biaoti"
                android:layout width="wrap content"
                android:layout height="wrap content"
                android:layout toRightOf="@+id/tupian"
                android:layout alignParentRight="true"
                android:layout alignParentTop="true"
                android:layout above="@+id/wenzi"
                android:layout alignWithParentIfMissing="true"
                android:gravity="center vertical"
                android:textSize="20dip" />
        <TextView android:id="@+id/wenzi"
                android:layout width="fill parent"
                android:layout height="wrap content"
                android:layout toRightOf="@+id/tupian"
                android:layout_alignParentBottom="true"
                android:layout alignParentRight="true"
                android:singleLine="true"
                android:ellipsize="marquee"
                android:textSize="15dip" />
    </RelativeLayout>
```

（2）打开 src/com.example.listimage/MainActivity.java 文件，修改并添加一些代码，代码清单如下。

代码清单：src/com.example.listimage/MainActivity.java

```
    package com.example.listimage;
    import java.util.ArrayList;
    import java.util.HashMap;
    import java.util.Map;
    import android.os.Bundle;
    import android.app.Activity;
    import android.view.Menu;
    import android.os.Bundle;
    import android.app.ListActivity;
    import android.view.View;
    import android.widget.AdapterView;
    import android.widget.AdapterView.OnItemClickListener;
    import android.widget.ArrayAdapter;
    import android.widget.ListView;
    import android.widget.SimpleAdapter;
    import android.widget.Toast;
    public class MainActivity extends ListActivity {
       private String[] mListTitle = { "伽利略", "培 根", " 拜 伦", " 狄更斯"," 马克思 "};
       private String[] mListStr = { "生命有如铁砧，愈被敲打，愈能发出火花。 ", " 瓜
       是长大在营养肥料里的最甜，天才是长在恶性土壤中的最好。", "悲观的人虽生犹死，乐观的
       人永生不老。", "顽强的毅力可以征服世界上任何一座高峰！", "生活就像海洋，只有意志
       坚强的人，才能到达彼岸。 " };
       ListView mListView = null;
       ArrayList<Map<String,Object>> mData= new ArrayList<Map<String,Object>>();
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        mListView = getListView();
        int lengh = mListTitle.length;
        for(int i =0; i < lengh; i++) {
            Map<String,Object> item = new HashMap<String,Object>();
            item.put("image", R.drawable.one);
            item.put("title", mListTitle[i]);
            item.put("text", mListStr[i]);
            mData.add(item);
        }
        SimpleAdapter adapter = new SimpleAdapter(this,mData,R.layout.
        activity main,
                new String[]{"image","title","text"},new int[]{R.id.tupian,
                R.id.biaoti,R.id.wenzi});
        setListAdapter(adapter);
        mListView.setOnItemClickListener(new OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> adapterView, View view,
            int position, long id) {
                Toast.makeText(MainActivity.this,"您喜欢的名言：" + mListTitle
                [position] + "-"+mListStr[position], Toast.LENGTH SHORT).show();
            }
        });
        super.onCreate(savedInstanceState);
    }
}
```

3.5　网格视图（GridView）

GridView 是按照行列的方式来显示内容的，一般用于显示图片等内容，如实现九宫格图，用 GridView 是首选，也是最简单的，主要用于设置 Adapter。

（1）Context：Context 提供了关于应用环境全局信息的接口。它是一个抽象类，它的执行被 Android 系统所提供。它允许获取以应用为特征的资源和类型。同时启动应用级的操作，如启动 Activity、broadcasting 和接收 intents。

（2）public void setAdapter (ListAdapter adapter)：设置 GridView 的数据，参数 adapter 为 grid 提供数据的适配器。

（3）public View getView(int position, View convertView, ViewGroup parent)各参数的含义如下。

- **position**　该视图在适配器数据中的位置。
- **convertView**　旧视图。
- **parent**　此视图最终会被附加到的父级视图。

（4）ImageView：显示任意图像，如图标。ImageView 类可以加载各种来源的图片（如资源或图片库），需要计算图像的尺寸，比便可以在其他布局中使用，并提供例如缩放和着

色（渲染）各种显示选项。

（5）public void setAdjustViewBounds (boolean adjustViewBounds)。

当需要在 ImageView 调整边框保持可绘制对象的比例时，将该值设为真。

（6）public void setScaleType (ImageView.ScaleType scaleType)。

控制图像应该如何缩放和移动，以使图像与 ImageView 一致。参数 scaleType 是需要的缩放方式。

案例：使用 GridView 编写一个界面，如图 3-8 所示。

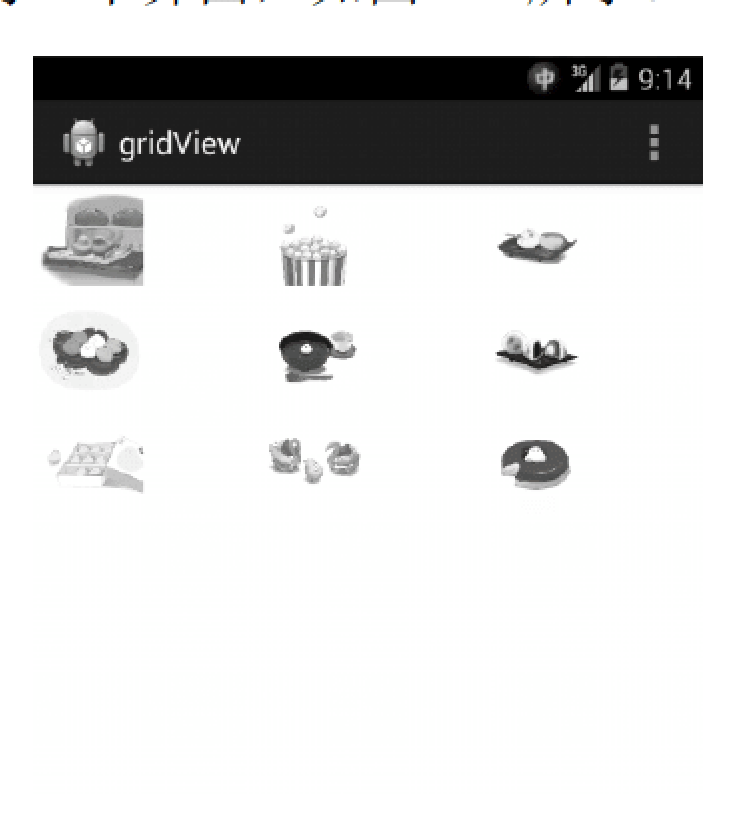


图 3-8　GridView 效果图

实现步骤如下。

（1）编写 activity_main.xml 布局文件，添加一个 GridView 控件，打开 res/layout/activity_main.xml 文件，修改并添加一些代码，代码清单如下。

代码清单：res/layout/activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout width="fill parent"
   android:layout height="fill parent"
   android:orientation="vertical" >

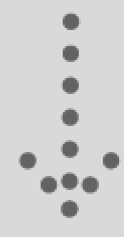
   <GridView
      android:id="@+id/GridViewone"
      android:layout width="wrap content"
      android:layout height="wrap content" >
   </GridView>

</LinearLayout>
```

（2）打开 src/com.example. gridview/MainActivity.java 文件，修改并添加一些代码，代码清单如下。

代码清单：src/com.example.gridview/MainActivity.java

```
package com.example.gridview;
import android.os.Bundle;
import android.app.Activity;
```



```
import android.content.Context;
import android.view.Menu;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.GridView;
import android.widget.ImageView;

public class MainActivity extends Activity {
    private GridView gv;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity main);
        gv=(GridView)findViewById(R.id.GridViewone);
      //设置 GridView 的列数
        gv.setNumColumns(3);
        //为 GridView 设置适配器
        gv.setAdapter(new MyAdapter(this));
    }
    ////自定义适配器
    class MyAdapter extends BaseAdapter{
    ////图片 ID 数组
    private Integer[] imgs = {
            R.drawable.one,
            R.drawable.two,
            R.drawable.three,
            R.drawable.four,
            R.drawable.five,
            R.drawable.six,
            R.drawable.seven,
            R.drawable.eight,
            R.drawable.nine,
    };
    ////上下文对象

    Context context;
    //构造方法
    MyAdapter(Context context){
        this.context = context;
    }


    //获得数量
        public int getCount() {
            //TODO Auto-generated method stub
            return imgs.length;
        }
        //获得当前选项
        public Object getItem(int position) {
```

```
            // TODO Auto-generated method stub
            return position;
        }
        //获得当前选项 ID
        public long getItemId(int position) {
            // TODO Auto-generated method stub
            return position;
        }
        //创建 View 方法
        public View getView(int position, View convertView, ViewGroup parent) {
            //TODO Auto-generated method stub
             ImageView imageView;
               if (convertView == null) {
                 //实例化 ImageView 对象
                   imageView = new ImageView(context);
                 //设置 ImageView 对象布局
                   imageView.setLayoutParams(new GridView.LayoutParams(125, 125));
                 //设置边界对齐
                   imageView.setAdjustViewBounds(false);
                 //设置刻度类型
                   imageView.setScaleType(ImageView.ScaleType.CENTER CROP);
                 //设置间距
                   imageView.setPadding(8, 8, 8, 8);
               } else {
                   imageView = (ImageView) convertView;
               }
                 //为 ImageView 设置图片资源
                   imageView.setImageResource(imgs[position]);
                   return imageView;
            }
            }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        //Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

}
```

3.6 控件的综合应用案例

案例描述：使用本章所学的常用控件编写一个注册界面，效果如图 3-9 所示。

案例分析：TextView、EditText、RadioButton、Button、ToggleButton、CheckBox、Spinner、imagebutton、imageview、Spinner 等控件，采用的布局方式是相对布局。



图 3-9　注册界面图

实现步骤如下：

打开 res/layout/activity_main.xml 布局文件，修改并添加一些代码，代码清单如下。

代码清单：res/layout/activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout width="match parent"
    android:layout height="match parent"
    android:paddingBottom="@dimen/activity vertical margin"
    android:paddingLeft="@dimen/activity horizontal margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity vertical margin"
    tools:context=".MainActivity" >

    <TextView
        android:id="@+id/textView1"
        android:layout width="wrap content"
        android:layout height="wrap content"
        android:layout alignParentTop="true"
        android:layout centerHorizontal="true"
        android:text="用户注册"
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <TextView
        android:id="@+id/textView2"
        android:layout width="wrap content"
        android:layout height="wrap content"
        android:layout below="@+id/textView1"
        android:layout_marginRight="24dp"
```

```
        android:layout marginTop="24dp"
        android:layout toLeftOf="@+id/textView1"
        android:text="用  户  名"
        android:textColor="#0000ff" />

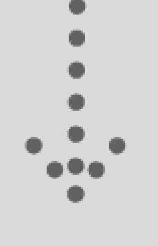
    <TextView
        android:id="@+id/TextView02"
        android:layout width="wrap content"
        android:layout height="wrap content"
        android:layout alignLeft="@+id/textView2"
        android:layout alignRight="@+id/textView2"
        android:layout below="@+id/textView2"
        android:layout marginTop="20dp"
        android:text="密     码"
        android:textColor="#ff00ff" />

    <TextView
        android:id="@+id/TextView01"
        android:layout width="wrap content"
        android:layout height="wrap content"
        android:layout alignLeft="@+id/TextView02"
        android:layout below="@+id/TextView02"
        android:layout marginTop="20dp"
        android:text="确认密码"
        android:textColor="#ff00ff" />

    <EditText
        android:id="@+id/editText3"
        android:layout width="wrap content"
        android:layout height="wrap content"
        android:layout alignLeft="@+id/editText2"
        android:layout below="@+id/editText2"
        android:ems="10"
        android:inputType="textPassword" />

    <EditText
        android:id="@+id/editText1"
        android:layout width="wrap content"
        android:layout height="wrap content"
        android:layout above="@+id/TextView02"
        android:layout alignLeft="@+id/editText2"
        android:ems="10" />

    <EditText
        android:id="@+id/editText2"
        android:layout width="wrap content"
        android:layout height="wrap content"
        android:layout below="@+id/editText1"
        android:layout toRightOf="@+id/TextView02"
        android:ems="10"
        android:inputType="textPassword"
```



```
        android:textColor="#ff00ff" />

    <RadioGroup
        android:id="@+id/radioGroup1"
        android:layout width="wrap content"
        android:layout height="wrap content"
        android:layout alignLeft="@+id/editText3"
        android:layout below="@+id/TextView01"
        android:layout marginLeft="24dp"
        android:layout marginTop="9dp"
        android:orientation="horizontal" >

        <RadioButton
            android:id="@+id/radioButton1"
            android:text="男" />

        <RadioButton
            android:id="@+id/radioButton2"
            android:layout width="wrap content"
            android:text="女" />
    </RadioGroup>

    <TextView
        android:id="@+id/textView3"
        android:layout width="wrap content"
        android:layout height="wrap content"
        android:layout below="@+id/TextView01"
        android:layout_marginTop="15dp"
        android:layout toLeftOf="@+id/radioGroup1"
        android:text="性      别" />

    <Spinner
        android:id="@+id/zwxz"
        android:layout width="wrap content"
        android:layout height="wrap content"
        android:layout alignLeft="@+id/radioGroup1"
        android:layout below="@+id/textView3"
        android:layout marginTop="10dp" />

    <TextView
        android:id="@+id/textView4"
        android:layout width="wrap content"
        android:layout height="wrap content"
        android:layout alignLeft="@+id/textView3"
        android:layout below="@+id/textView3"
        android:layout marginTop="15dp"
        android:text="职位" />

    <TableRow
        android:id="@+id/tableRow1"
        android:layout_width="wrap_content"
```

```
        android:layout height="wrap content"
        android:layout alignLeft="@+id/zwxz"
        android:layout below="@+id/textView4"
        android:layout marginTop="10dp" >

        <TableRow
            android:layout width="wrap content"
            android:layout height="wrap content" >

            <CheckBox
                android:id="@+id/swim"
                android:layout width="wrap content"
                android:layout height="wrap content"
                android:text="swim" />

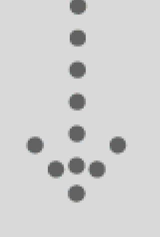
            <CheckBox
                android:id="@+id/read"
                android:layout width="wrap content"
                android:layout height="wrap content"
                android:text="read" />
        </TableRow>
    </TableRow>

    <TextView
        android:id="@+id/textView5"
          android:layout marginTop="10dp"
        android:layout width="wrap content"
        android:layout_height="wrap_content"
        android:layout alignLeft="@+id/textView4"
        android:layout alignTop="@+id/tableRow1"
        android:text="爱      好" />

    <ToggleButton
        android:id="@+id/toggleButton1"

        android:layout width="wrap content"
        android:layout height="wrap content"
        android:layout below="@+id/tableRow1"
        android:layout centerHorizontal="true"
        android:textOn="是"
        android:textOff="不是"
        android:text="ToggleButton" />

    <TextView
        android:id="@+id/textView6"
        android:layout width="wrap content"
        android:layout height="wrap content"
        android:layout alignBaseline="@+id/toggleButton1"
        android:layout alignBottom="@+id/toggleButton1"
        android:layout alignLeft="@+id/textView5"
        android:text="你喜欢编程吗" />
```



```
    <ImageView
        android:id="@+id/imageView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/toggleButton1"
        android:layout_marginTop="16dp"
        android:layout_toRightOf="@+id/radioGroup1"
        android:src="@drawable/yzm" />

    <TextView
        android:id="@+id/textView7"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/imageView1"
        android:layout_alignLeft="@+id/textView6"
        android:text="验证码" />

    <EditText
        android:id="@+id/editText4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/imageView1"
        android:layout_alignLeft="@+id/editText3"
        android:layout_alignRight="@+id/textView1"
        android:ems="10" >

        <requestFocus />
    </EditText>

    <ImageButton
        android:id="@+id/imageButton1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/textView7"
        android:layout_marginTop="38dp"
        android:layout_toRightOf="@+id/textView4"
        android:src="@drawable/zc" />

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/imageButton1"
        android:layout_alignTop="@+id/imageButton1"
        android:layout_toRightOf="@+id/editText4"
        android:textSize="15px"
        android:text="取消" />

</RelativeLayout>
```

（2）打开 src/com.example. gridview/MainActivity.java 文件，实现 Spinner 控件的功能，修改并添加一些代码，代码清单如下。

代码清单：src/com.example. gridview/MainActivity.java

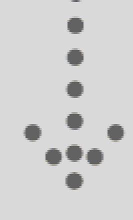
```
import android.os.Bundle;
import android.app.Activity;
import android.view.Menu;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.Toast;

public class MainActivity extends Activity {
  private Spinner zwxz;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity main);
        zwxz = (Spinner) findViewById(R.id.zwxz);
    String[] a = { "CEO", "CFO", "PM" };
    ArrayAdapter A = new ArrayAdapter(this,
        android.R.layout.simple spinner item, a);
    zwxz.setAdapter(A);
    }
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

}
```

3.7 本章小结

本章主要以案例形式讲述了 ImageButton 控件、ImageView 控件、RadioButton 控件、CheckBox 控件、ListView 控件和 GridView 控件的属性及如何使用，最后介绍了一个控件的综合应用案例。对于初学者来说有一定帮助，请在开发工具中多调试本章的案例。



第 4 章　菜单和对话框

菜单是用户界面中最常见的元素之一，使用非常频繁，在手机应用程序中，由于受到手机屏幕大小的制约，菜单在手机应用中的使用减少很多，但是依然有手机应用程序会添加菜单。当图形用户界面在前台运行时，如果用户按下手机上的 Menu 键，就会在屏幕底端弹出相应的选项菜单，但其对应的功能是需要程序开发者编程实现的。如果在应用程序开发中没有实现其功能，则在程序运行时按下手机上的 Menu 键是不会有作用的。在 Android 中，菜单被分为如下三种，选项菜单（OptionsMenu）、上下文菜单（ContextMenu）和子菜单（SubMenu）。

4.1　选项菜单和子菜单

一个 Menu 对象代表一个菜单，可以添加菜单项 MenuItem，也可以添加子菜单 SubMenu。Android 中的菜单是显示在 Activity 之上的元素，分为 OptionsMenu、Context Menu、SubMenu 等多种类型。通常通过回调方法来创建菜单并处理菜单按下的事件。其中，SubMenu 代表一个普通菜单，可以包含一到多个菜单项。Context Menu 代表一个子菜单，由一到多个菜单项组成。

Menu 类中定义了若干个 add()和 addSubMenu()方法，其中，add()用于添加菜单项，addSubMenu()用于添加子菜单，这些重载方法的区别在于是否将子菜单、菜单项添加到指定菜单中、是否使用资源文件中的字符串资源来设置标题等。

SubMenu 继承了 Menu，它代表了一个子菜单，除了 Menu 的方法外还有设置菜单头图标的方法 SetHeaderIcon(Drawable icon)、设置菜单头标题的方法 SetHeaderTitle(int titleRes)和使用 View 来设置菜单头的方法 SetHeaderView(View view)等设置属性的方法。

在应用程序中要添加菜单或者子菜单的步骤如下。

（1）需要重写 Activity 的 onCreateOptionsMenu(Menu menu)方法，在该方法中创建菜单，添加其菜单项或者子菜单。

（2）对其触发的事件进行监听。如果希望应用程序能够响应菜单项的单击事件重写 Activity 的 onOptionsItemSelected(MenuItem mi)，这样才能够实现根据不同的菜单选项执行不同操作，还可以用户自定义菜单项的监听器。

4.1.1　创建 OptionsMenu 菜单实例

选项菜单 OptionsMenu 默认样式是在屏幕底部弹出一个菜单， 其实现方式有两种：第一种是通过 Menu 类在创建菜单；第二种是通过 XML 文件布局文件添加菜单的样式。下面通过实例展示如何使用两种方式创建选项菜单。

案例：使用 OptionsMenu 设计一个界面，效果如图 4-1 所示。当选择了某个选项时，显示一个提示信息。



图 4-1　OptionsMenu 效果图

方法一：通过 Menu 类来创建菜单。

重载 onCreateOptionsMenu(Menu menu)方法，并在此方法中通过 Menu 类的 add()方法添加菜单项该方法的四个参数，依次是代表组别、Id、显示顺序和菜单的显示文本。其实现步骤如下。

（1）创建项目 MenuDemo，并创建一个名字为 DefaultMenu 的 Activity。

（2）重载 onCreateOptionsMenu(Menu menu)方法，并在此方法中添加菜单项，最后返回 true，如果 false，菜单则不会显示。

（3）使用 onOptionsItemSelected(MenuItem item)方法为菜单项注册事件。

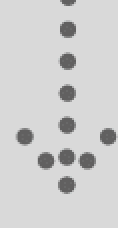
（4）其他按需要重载。如重载 onOptionsMenuClosed(Menu menu)可以处理菜单关闭后发生的动作等。

```
package com.chapt5;

import android.app.Activity;
import android.os.Bundle;

import android.view.Menu;
import android.view.MenuItem;
import android.widget.Toast;

public class DefaultMenu extends Activity {
       /** Called when the activity is first created. */
       public void onCreate(Bundle savedInstanceState) {
```



```
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
    }
    public boolean onCreateOptionsMenu(Menu menu) {
           /* * * add()方法的四个参数，依次是:
            * 1. 组别，如果不分组的话就写 Menu.NONE, * *
            * 2. Id，这个很重要，Android 根据这个 Id 来确定不同的菜单 *
            * 3. 顺序，那个菜单现在在前面由这个参数的大小决定 * *
            * 4. 文本，菜单的显示文本 */
            menu.add(Menu.NONE, Menu.FIRST + 1, 5, "删除").setIcon(
                          android.R.drawable.ic menu delete);
            // setIcon()方法为菜单设置图标，这里使用的是系统自带的图标
            menu.add(Menu.NONE, Menu.FIRST + 2, 2, "保存").setIcon(
                          android.R.drawable.ic menu edit);
            menu.add(Menu.NONE, Menu.FIRST + 3, 6, "帮助").setIcon(
                          android.R.drawable.ic menu help);
            menu.add(Menu.NONE, Menu.FIRST + 4, 1, "添加").setIcon(
                          android.R.drawable.ic menu add);
            menu.add(Menu.NONE, Menu.FIRST + 5, 4, "详细").setIcon(
                          android.R.drawable.ic menu info details);
            menu.add(Menu.NONE, Menu.FIRST + 6, 3, "发送").setIcon(
                          android.R.drawable.ic menu send);
            return true;
     }

     public boolean onOptionsItemSelected(MenuItem item) {
            switch (item.getItemId()) {
            case Menu.FIRST + 1:
                   Toast.makeText(this, "删除菜单被点击了", Toast.
                   LENGTH LONG).show();break;
            case Menu.FIRST + 2:
                   Toast.makeText(this, "保存菜单被点击了", Toast.
                   LENGTH LONG).show();
                   break;
            case Menu.FIRST + 3:
                   Toast.makeText(this, "帮助菜单被点击了", Toast.
                   LENGTH_LONG).show();
                   break;
            case Menu.FIRST + 4:
                   Toast.makeText(this, "添加菜单被点击了", Toast.
                   LENGTH LONG).show();
                   break;
            case Menu.FIRST + 5:
                   Toast.makeText(this, "详细菜单被点击了", Toast.
                   LENGTH LONG).show();
                   break;
            case Menu.FIRST + 6:
                   Toast.makeText(this, "发送菜单被点击了", Toast.
                   LENGTH LONG). show();
                   break;
            }
```

```
                return false;
        }

        public void onOptionsMenuClosed(Menu menu) {
                Toast.makeText(this, "选项菜单关闭了", Toast.LENGTH_LONG).
                show();
        }
}
```

方法二：通过 XML 布局实现菜单。

通过 Layout 布局创建菜单对应的 xml 文件，然后在 onCreateOptionsMenu 中设置 menu 为定义的 res/menu/menu.xml，其具体实现步骤如下。

（1）创建项目 MenuDemo。

（2）在 string.xml 文件中输入需要显示的字符，打开 res/layout/strings.xml 文件。

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
        <string name="hello">Hello World, DefaultMenu!</string>
        <string name="app_name">menudemo</string>
        <string name="add">添加</string>
        <string name="save">保存</string>
        <string name="send">发送</string>
        <string name="detail">详情</string>
        <string name="delete">删除</string>
        <string name="help">帮助</string>
</resources>
```

（3）选择“新建”→“Other” →“Android” →“Android XML File”，创建名为“menu.xml”文件。

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
       <item android:id="@+id/add" android:title="@string/add"
             android:icon="@android:drawable/ic_menu_add" />
       <item android:id="@+id/sava" android:title="@string/save"
             android:icon="@android:drawable/ic_menu_save" />
       <item android:id="@+id/send" android:title="@string/send"
             android:icon="@android:drawable/ic_menu_send" />
       <item android:id="@+id/detail" android:title="@string/detail"
             android:icon="@android:drawable/ic_menu_info_details" />
       <item android:id="@+id/delete" android:title="@string/delete"
             android:icon="@android:drawable/ic_menu_delete" />
       <item android:id="@+id/help" android:title="@string/help"
             android:icon="@android:drawable/ic_menu_help" />
</menu>
```

（4）创建一个名字为 DefaultMenu 的 Activity，并重载 onCreateOptionsMenu(Menu menu)方法，并使用 onOptionsItemSelected(MenuItem item)方法为菜单项注册事件。

```
package com.chapt5;
```

```
import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.widget.Toast;
public class DefaultMenu extends Activity {
      public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
      }
      public boolean onCreateOptionsMenu(Menu menu) {
            MenuInflater inflater = getMenuInflater();
            inflater.inflate(R.menu.menu, menu);
            return true;
      }
      public boolean onOptionsItemSelected(MenuItem item) {
            switch (item.getItemId()) {
            case R.id.delete:
                  Toast.makeText(this, "删除菜单被点击了", Toast.LENGTH_
                  LONG).show();
                  break;
            case R.id.sava:
                  Toast.makeText(this, "保存菜单被点击了", Toast.LENGTH_
                  LONG).show();
                  break;
            case R.id.help:
                  Toast.makeText(this, "帮助菜单被点击了", Toast.LENGTH
                  LONG).show();
                  break;
            case R.id.add:
                  Toast.makeText(this, "添加菜单被点击了", Toast.LENGTH
                  LONG).show();
                  break;
            case R.id.detail:
                  Toast.makeText(this, "详细菜单被点击了", Toast.LENGTH_
                  LONG).show();
                  break;
            case R.id.send:
                  Toast.makeText(this, "发送菜单被点击了", Toast.LENGTH_
                  LONG).show();
                  break;
            }
            return false;
      }
      public void onOptionsMenuClosed(Menu menu) {
            Toast.makeText(this, "选项菜单关闭了", Toast.LENGTH_
            LONG).show();
      }
}
```

知识点：

（1）在用 XML 布局来实现时，在 onCreateOptionsMenu 中，需要通过 MenuInflater 对象把 menu XML 文件转化为 menu 对象，通过 Menu.add()方法则直接添加菜单项来构建菜单。

（2）Toast 是 Android 中用来显示信息的一种机制，和 Dialog 不同的是，Toast 是没有焦点的，而且 Toast 显示的时间有限，过一定的时间就会自动消失。

4.1.2 监听菜单事件

除了重写 onOptionsItemSelected(MenuItem item)方法处理菜单的单击事件外，可以通过菜单项的 setOnMenuItemClickListener 方法为不同的菜单项分别绑定监听器。采用这种方式无需为每个菜单项指定 ID，而是通过获取所添加的 MenuItem 对象，然后给对象绑定监听者。以下给出“delete”菜单项的创建、事件注册与响应。由于篇幅有限，其他的实现与其相同。

```
public boolean onCreateOptionsMenu(Menu menu) {
      MenuItem deleteItem = menu.add(Menu.NONE, Menu.FIRST + 1, 5,
      "删除").setIcon(android.R.drawable.ic_menu_delete);
      deleteItem.setOnMenuItemClickListener(new
      OnMenuItemClickListener () {
      public boolean onMenuItemClick(MenuItem arg0) {
      Toast.makeText(MenuDemo2.this, "删除菜单被点击了", Toast.LENGTH_
      LONG).show();
      return false;
      }
      });
   return true;
}
```

说明：（1）该方法实现效果与前一个完全相同，区别仅在于处理菜单事件的监听方式不同，一般来说，通过重载 onOptionsItemSelected(MenuItem item)方法处理菜单的单击事件更加简洁，因为所有的事件处理代码都控制在该方法内，通过绑定事件监听器使程序具有更清晰的逻辑性，但是代码显得有些臃肿。

（2）如果是通过 XML 布局文件来实现的菜单，可以通过 MenuItem delete=(MenuItem)findViewById(R.id.delete)语句获取菜单项对象。

（3）如果希望所创建的菜单项是单选菜单项或多选菜单项，则可以调用菜单项的 setCheckable(Boolean chackable)来设置该菜单项是否可以被勾选。通过调用 setGroupCheckable()设置组里的菜单是否可勾选。

（4）可以通过菜单项的 setShortcut()方法为其设置快捷键。

4.1.3 与菜单项关联的 Activity 的设置

在应用程序中如果需要单击某个菜单项来启动其他 Activity 或者 Service 时，不需要开

发者编写任何事件处理代码，只要调用 MenuItem 的 setIntent(Intent intent)方法即可。该方法实现把菜单项与指定的 Intent 关联在一起，当用户单击该菜单项时，该 Intent 所代表的组件将会被启动。

案例：通过菜单项启动另一个 Activity，效果如图 4-2 所示。当选择了“Start the other Activity”时，启动另一个 Activity。



图 4-2　通过菜单项关联 Activity 效果图

（1）创建项目 menuitemactivity。

（2）创建 MenuitemActivity，并重载 onCreate()和 onCreateOptionsMenu ()方法。

```
package com.chapt5;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
public class MenuitemActivity extends Activity {
      public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
      }
      public boolean onCreateOptionsMenu(Menu menu) {
            MenuItem mi = menu.add("Start the other Activity");
            mi.setIntent(new Intent(this, OtherActivity.class));
            return super.onCreateOptionsMenu(menu);
      }
}
```

（3）创建 OtherActivity。并在 AndroidManifest.xml 中注册，在其中添加：

```
<activity android:name=".OtherActivity"
                  android:label="This is Other Activity!" />
```



4.2 上下文菜单

Android 用 ContextMenu 来代表上下文菜单，类似于桌面程序的右键弹出式菜单，在 Android 中不是通过用户右击鼠标而得到，而是通过长时间按住界面上的元素得到事先设计好的上下文菜单。开发上下文菜单的方法与选项菜单的方法基本相似，因为 ContextMenu 也是 Menu 的子类，所以可用相同的方法为它添加菜单项。其区别在于：开发上下文菜单不是重写 onCreateOptionsMenu(Menu menu) 方法，而是调用 onCreate ContextMenu (ContextMenu menu, View source,ContextMenu.ContextMenuIfno menuInfo)方法，该方法在每次启动上下文菜单时都会被调用一次，在该方法中可以通过使用 add()方法添加相应的菜单项。

开发上下文菜单的步骤如下。

（1）重写 onCreateContextMenu()方法。

（2）调用 Activity 的 registerForContextMenu(View view)为 view 组件注册上下文菜单。

（3）重载 onContextItemSelected(MenuItem mi)或者绑定事件监听器，对菜单项进行事件相应。

案例：定义上下文菜单，让用户进行颜色选择，根据用户所选颜色的不同来更改文本框的背景颜色，效果如图 4-3 所示。

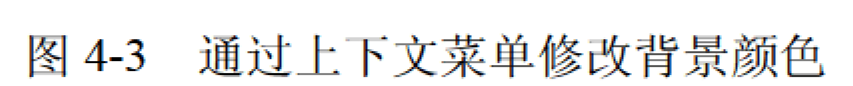


图 4-3 通过上下文菜单修改背景颜色

（1）创建项目 ContextMenu。

（2）创建 ContextMenuActivity，并在 onCreate()方法中通过方法为文本框注册上下文菜单。

（3）重载 onCreateContextMenu ()在该方法中创建含有“红色”、“绿色”、“蓝色”和“退出”四个菜单项的菜单。

（4）重载 onContextItemSelected()方法对事件进行注册。

```
package com.chapt5;
import android.app.Activity;
import android.graphics.Color;
import android.os.Bundle;
import android.view.ContextMenu;
import android.view.ContextMenu.ContextMenuInfo;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.TextView;
public class ContextMenuActivity extends Activity {
      TextView text;
      public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
```



```
            text = (TextView) findViewById(R.id.contextmenu);
            registerForContextMenu(text);// 为文本框注册上下文菜单
        }
        // 每次创建上下文菜单时都会触发该方法
        public void onCreateContextMenu(ContextMenu menu, View v,
                ContextMenuInfo menuInfo) {
            text.setText("这是关于上下文菜单的测试操作！");// 设置提示信息
            menu.add(1, Menu.FIRST + 1, 1, "红色");
            menu.add(1, Menu.FIRST + 2, 2, "绿色");
            menu.add(1, Menu.FIRST + 3, 3, "蓝色");
            menu.add(1, Menu.FIRST + 4, 4, "退出");
            menu.setGroupCheckable(1, true, true);
            menu.setHeaderTitle("请选择：");
            menu.setHeaderIcon(android.R.drawable.ic dialog info);
        }
        public boolean onContextItemSelected(MenuItem item) {
            switch (item.getItemId()) {
            case Menu.FIRST + 1:
                text.setBackgroundColor(Color.RED);
                break;
            case Menu.FIRST + 2:
                text.setBackgroundColor(Color.GREEN);
                break;
            case Menu.FIRST + 3:
                text.setBackgroundColor(Color.BLUE);
                break;
            case Menu.FIRST + 4:
                ContextMenuActivity.this.finish();
                break;
            default:
                item.setChecked(true);
            }
            return true;
        }
    }
```

4.3 Android 中对话框

Android 中实现对话框可以自定义对话框，同时 Android 也提供了丰富的对话框支持，常用的对话框有下面 4 种。

（1）AlertDialog：功能丰富、应用最广泛；

（2）ProgressDialog：进度对话框，该对话框只对简单进度条封装；

（3）DatePickerDialog：日期选择对话框，该对话框是对 DatePicker 的包装；

（4）TimePickerDialog：时间选择对话框，是对 TimePicker 的包装，这四种对话框中功能最强、用法最灵活的就是 AlertDialog，因此，它的应用最为广泛。

4.3.1 提示对话框 AlertDialog

AlertDialog 是一个提示窗口，要求用户做出选择，该对话框中一般会有几个选择按钮、标题信息和提示信息。AlertDialog 提供了一些方法来生成四种预定义对话框。

（1）带消息、带 N 个按钮的提示对话框。

（2）带列表、带 N 个按钮的列表对话框。

（3）带多个单选列表项，带 N 个按钮的对话框。

（4）带多个多选列表项，带 N 个按钮的对话框。

AlertDialog 的构造方法全部是 Protected 的，所以不能直接通过 AlertDialog 对象来创建对话框。要创建一个 AlertDialog，就要用到 AlertDialog.Builder 中的 create()方法。使用 AlertDialog.Builder 创建对话框需要了解以下几个方法：（1）setTitle()为对话框设置标题；（2）setIcon()为对话框设置图标；（3）setMessage()为对话框设置内容；（4）setView()给对话框设置自定义样式；（5）setItems()设置对话框要显示的一个 list，一般用于显示几个命令；（6）setMultiChoiceItems()来设置对话框显示一系列的复选框；（7）setNeutralButton()普通按钮；（8）setPositiveButton()对话框添加"Yes"按钮；（9）setNegativeButton()对话框添加"No"按钮；（10）create ()创建对话框；（11）show ()显示对话框。

创建 AlertDialog 的主要步骤如下。

（1）获得 AlertDialog 的静态内部类 Builder 对象，由该类创建对话框；

（2）通过 Builder 对象设置对话框的标题、按钮及按钮将要响应的事件；

（3）调用 Builder 对象的 create()方法创建对话框；

（4）调用 AlertDialog 的 show()方法显示对话框。

案例：创建不同类型的对话框，其运行效果如以下各图所示，所有的对话框定义在 Activitiy 中，本例只给出代码片段。

1）内容输入框

```
new AlertDialog.Builder(this).setTitle("请输入").setIcon(
                android.R.drawable.ic dialog info).setView(
                new EditText(this)).setPositiveButton("确定", null)
                .setNegativeButton("取消", null).show();
```

其运行效果如图 4-4 所示。



图 4-4　内容输入对话框



2）带按钮的提示对话框

```
Dialog alertDialog = new AlertDialog.Builder(this).setTitle("提示")
      .setMessage("您确定退出吗？"). setIcon(R.drawable.icon)
      .setPositiveButton("确定", new DialogInterface.OnClickListener() {
         public void onClick(DialogInterface dialog, int which) {
            dialog.dismiss();
            DialogDemoActivity.this.finish();
                        }
                        })
   .setNegativeButton("取消", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int which) {
            dialog.dismiss();
            }
            }).create();
        alertDialog.show();
```

其运行效果如图 4-5 所示。

3）列表对话框

用 setItems(CharSequence[] items, final OnClickListener listener)方法来实现类似 ListView 的 AlertDialog 第一个参数是要显示的数据的数组，第二个参数是单击某个 item 的触发事件

```
final String[] arrayFruit = new String[] { "苹果", "橘子", "草莓", "香蕉" };
Dialog alertDialog = new AlertDialog.Builder(this).setTitle("你喜欢吃哪种水果？").setIcon(R.drawable.icon)
       .setItems(arrayFruit, new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int which) {
           Toast.makeText(DialogDemoActivity.this, arrayFruit[which],
           Toast.LENGTH SHORT).show();
                 }})
      .setNegativeButton("取消", new DialogInterface.OnClickListener() {
         public void onClick(DialogInterface dialog, int which) {
                        }
                        }).create();
      alertDialog.show();
```

其运行效果如图 4-6 所示。

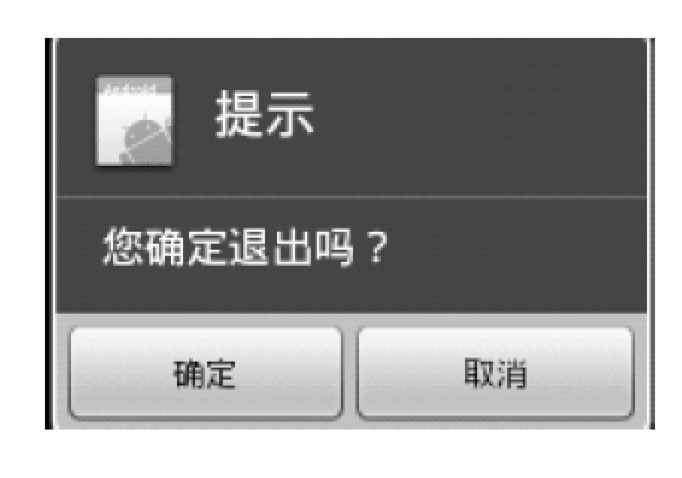


图 4-5　带按钮的提示对话框



图 4-6　列表对话框



4）单选列表对话框

用 setSingleChoiceItems(CharSequence[] items, int checkedItem, final OnClickListener listener)方法来实现类似 RadioButton 的 AlertDialog。第一个参数是要显示数据的数组，第二个参数是初始值（初始被选中的 item），第三个参数是单击某个 item 的触发事件。

```
final String[] arrayFruit = new String[] { "苹果", "橘子", "草莓", "香蕉" };
Dialog alertDialog = new AlertDialog.Builder(this).setTitle("你喜欢吃哪种水果? ").setIcon(R.drawable.icon)
      .setSingleChoiceItems(arrayFruit, 0,
      new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog,int which) {
                              selectedFruitIndex = which; }
  }).setPositiveButton("确认", new DialogInterface.OnClickListener() {
      public void onClick(DialogInterface dialog, int which) {
            Toast.makeText(DialogDemoActivity.this, arrayFruit
            [selectedFruitIndex], Toast.LENGTH SHORT).show();}
                     })
      .setNegativeButton("取消", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int which) {
                  }
                  }).create();
      alertDialog.show();
```

其运行效果如图 4-7 所示。

图 4-7　单选列表对话框

5）多选列表对话框

用 setMultiChoiceItems(CharSequence[] items, boolean[] checkedItems, final OnMultiChoiceClickListener listener)方法来实现类似 CheckBox 的 AlertDialog。第一个参数是要显示的数据的数组，第二个参数是选中状态的数组，第三个参数是单击某个 item 的触发事件。

```
final String[] arrayFruit = new String[] { "苹果", "橘子", "草莓", "香蕉" };
final boolean[] arrayFruitSelected = new boolean[] { true, true,
false,false };
```

```
Dialog alertDialog = new AlertDialog.Builder(this).setTitle("你喜欢吃哪种水
果？").setIcon(R.drawable.icon)
        .setMultiChoiceItems(arrayFruit, arrayFruitSelected,
         new DialogInterface.OnMultiChoiceClickListener() {
         public void onClick(DialogInterface dialog,int which,
         boolean isChecked) {
              arrayFruitSelected[which] = isChecked;
              }
}).setPositiveButton("确认", new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int which) {
        StringBuilder stringBuilder = new StringBuilder();
        for (int i = 0; i < arrayFruitSelected.length; i++) {
           if (arrayFruitSelected[i] == true) {
              stringBuilder.append(arrayFruit[i] + "、");
               }
              }
        Toast.makeText(DialogDemoActivity.this,
        stringBuilder.toString(), Toast.LENGTH SHORT).show();
              }
        }).setNegativeButton("取消", new DialogInterface.
        OnClickListener() {
        public void onClick(DialogInterface dialog, int which) {
               }
             }).create();
        alertDialog.show();
```

其运行效果如图 4-8 所示。

6）自定义对话框

有时不能满足系统自带的 AlertDialog 风格，用户可以自己定义对话框。

案例：创建用户登录对话框，有用户名和密码，其运行效果如图 4-9 所示。



图 4-8　多选列表对话框



图 4-9　自定义对话框

操作步骤如下。

（1）创建项目 DialogDemo 和名为 DialogDemoActivity 的 Activity。

（2）创建 Login 画面的布局文件 login.xml。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout width="match parent"
    android:layout height="match parent"
    android:orientation="vertical">
<LinearLayout
    android:layout_width="fill_parent"
    android:layout height="wrap content"
    android:gravity="center">
<TextView android:layout width="0dip"
          android:layout height="wrap content"
          android:layout weight="1"
          android:text="@string/username" />
<EditText android:layout_width="0dip"
          android:layout height="wrap content"
          android:layout_weight="1" />
      </LinearLayout>
<LinearLayout
    android:layout width="fill parent"
    android:layout height="wrap content"
    android:gravity="center">
<TextView android:layout width="0dip"
          android:layout height="wrap content"
          android:layout weight="1"
          android:text="@string/password" />
<EditText android:layout width="0dip"
          android:layout height="wrap content"
          android:layout weight="1" />
     </LinearLayout>
</LinearLayout>
```

（3）重载 DialogDemoActivity 的 onCreate 方法，并添加代码。

```
LayoutInflater layoutInflater = LayoutInflater.from(this);
View myLoginView = layoutInflater.inflate(R.layout.login, null);
Dialog alertDialog = new AlertDialog.Builder(this).setTitle("用户登录").
                 setIcon(R.drawable.ic launcher).
                 setView(myLoginView).
                 setPositiveButton("登录", new DialogInterface.
                 OnClickListener() {
 public void onClick(DialogInterface dialog, int which) {
                 }
  }).setNegativeButton("取消", new DialogInterface.OnClickListener() {
public void onClick(DialogInterface dialog, int which) {
                }
               }).create();
 alertDialog.show();
```






4.3.2 进度对话框 ProgressDialog

ProgressDialog 类继承自 AlertDialog 类；同样存放在 android.app 包中。ProgressDialog 有两种形式：一种是圆圈旋转形式；另一种是水平进度条形式，可以通过属性设置来修改其形式。开发者可以通过该类提供的一系列的 set 方法，设置对话框中进度条的风格、进度条的最大值等属性。

案例：在主界面上放置一个命令按钮，当单击命令按钮时，弹出一个进度对话框，提示后台程序正在执行，稍等片刻，其运行效果如图 4-10 所示。



图 4-10 进度对话框运行效果图

其操作步骤如下。

（1）创建项目 ProgressDialogDemo 和名为 ProgressDialogActivity 的 Activity。

（2）修改 main.xml，其上放置一个命令按钮，其 id 为 button，text 为 android:text="@string/execute"。

（3）修改 string.xml，其内容如下。

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">ProgressDialogDemo</string>
    <string name="execute">执行</string>
    <string name="str_dialog_title">请稍等片刻</string>
    <string name="str_dialog_body">正在执行...</string>
</resources>
```

（4）修改 ProgressDialogActivity，其内容如下。

```
package com.chapt5;
import android.app.Activity;
import android.app.ProgressDialog;
```

```
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
public class ProgressDialogActivity extends Activity {
   private Button button=null;
   public ProgressDialog dialog=null;
   public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      button=(Button)findViewById(R.id.button);
    button.setOnClickListener(new OnClickListener(){
        public void onClick(View v) {
    String title =ProgressDialogActivity.this.getString(R.string.str
    dialog title);
      String body =ProgressDialogActivity.this.getString(R.string.str
      dialog_body);
    //显示 Progress 对话框
    dialog=ProgressDialog.show(ProgressDialogActivity.this,strDialogTitle,
    strDialogBody,true);new Thread()       {
          public void run(){
             try{
                //表示后台运行的代码段，以暂停 3 秒代替
                sleep(3000);
                }catch (InterruptedException e){
                    e.printStackTrace();
                    }finally{
                      //卸载 dialog 对象
                     dialog.dismiss();
                      }
                  }
                    }.start();
                    }
             });
    }
}
```

4.3.3 DatePickerDialog 和 TimePickerDialog

在 Android 应用中，DatePickerDialog 与 TimePickerDialog 分别表示日期对话框和时间对话框，都是以弹出式对话框形式出现的，使用方法基本相同。前者需要实现 OnDateSetListener 接口中的 onDateSet 方法，后者需要实现 OnTimeSetListener 接口中的 onTimeSet 方法，操作步骤如下。

（1）创建 DatePickerDialog 或 TimePickerDialog 对象，通过它们的 show()方法将其显示出来。

（2）为日期或时间对话框对象绑定监听者。

案例：在主界面上放置两个命令按钮“显示日期”和“显示时间”。当单击“显示日期”命令按钮时，弹出一个日期显示对话框；当单击“显示时间”命令按钮时，弹出一个



时间显示对话框，供用户进行选择。并将显示结果在界面的文本框上显示出来，其运行效果如图 4-11 所示。



图 4-11　日期时间对话框运行效果图

操作步骤如下。

（1）创建项目 DataTimePickerDemo 和名为 DataTimePickerActivity 的 Activity。

（2）修改 string.xml，其内容如下。

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">DataTimePickerDialogDemo</string>
    <string name="showdate">显示日期</string>
    <string name="showtime">显示时间</string>
</resources>
```

（3）修改布局文件 main.xml，其内容如下。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
 <LinearLayout
 android:orientation="horizontal"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    >
    <Button
    android:id="@+id/showdate"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
        android:text="@string/showdate"
        />
        <Button
        android:id="@+id/showtime"
        android:layout width="wrap content"
        android:layout height="wrap content"
        android:text="@string/showtime"
        />
        </LinearLayout>
        <EditText android:id="@+id/show"
      android:layout width="fill parent"
      android:layout height="wrap content"
      android:editable="false"
      android:cursorVisible="false"
      />
    </LinearLayout>
```

（4）对 DataTimePickerActivity 进行重新定义。

```
    public class DataTimePickerActivity extends Activity {
        public void onCreate(Bundle savedInstanceState) {
              super.onCreate(savedInstanceState);
              setContentView(R.layout.main);
              Button bt1 = (Button) findViewById(R.id.showdate);
              bt1.setOnClickListener(new ClickLis());
              Button bt2 = (Button) findViewById(R.id.showtime);
              bt2.setOnClickListener(new ClickLis());
        }
        class ClickLis implements OnClickListener {
           public void onClick(View v) {
              Calendar c = Calendar.getInstance();
              if (v.getId() == R.id.showdate) {
                 new DatePickerDialog(DataTimePickerActivity.this,
              new DatePickerDialog.OnDateSetListener() {
              public void onDateSet(DatePicker view, int year,
                 int monthOfYear, int dayOfMonth) {
              EditText show = (EditText) findViewById(R.id.show);
              show.setText("您选择的日期为: " + year + "年" + monthOfYear
              + "月" + dayOfMonth+ "日  ");
              }}, c.get(Calendar.YEAR), c.get (Calendar.MONTH),
             c.get(Calendar.DAY OF MONTH)).show();
              }
             else if (v.getId() == R.id.showtime) {
              new TimePickerDialog(DataTimePickerActivity.this,
                    new TimePickerDialog.OnTimeSetListener() {
                    public void onTimeSet(TimePicker view,
                           int hourOfDay, int minute) {
                    EditText show = (EditText) findViewById(R.id.show);
                    show.setText("您选择的时间为: " + hourOfDay + "点"
                    + minute + "分");}}, c.get(Calendar.HOUR OF DAY),
                   c.get(Calendar.MINUTE),false).show();
```

```
                }
            }
        }
    }
```

4.4 提示信息

在某些情况下需要向用户弹出提示消息，如显示错误信息或收到短消息等，Android 提供弹出消息的方式、状态栏的提醒机制等方式提示信息。下面对弹出式提示信息 Toast 和 Notification 进行简单介绍。

4.4.1 Toast

Toast 是 Android 中用来显示提示信息的一种机制，这个提示信息框用于向用户生成简单的提示信息。与对话框不同的是 Toast 没有焦点，显示的时间有限，信息浮动显示设定的时长后会自动消失。创建 Toast 的一般步骤如下。

（1）调用 Toast 的构造器或静态方法 markText()创建一个 Toast 对象。

（2）调用 Toast 的方法设置该消息提示的对齐方式、显示内容、显示时长等属性。

（3）调用 Toast 的 show()方法将其显示出来。

Toast 一般用于显示简单的提示信息，如果需要显示较为复杂的信息，如图片、列表等，一般用对话框来完成，也可以用 Toast 的 setView(view)添加 view 组件的方式来实现，该方法允许用户自定义显示内容。创建 Toast 常用的方法如下。

```
    Toast t = Toast.makeText(Context,msg,Toast.LENGTH_SHORT 或 LENGTH_LONG);
```

例如，在运行中弹出一个 Toast，其提示信息为“你的愿望能实现”。

```
    Toast.makeText(getApplicationContext(), "你的愿望能实现", Toast.LENGTH
SHORT). show()
```

4.4.2 Notification

Notification 是 Android 提供的在状态栏的提醒机制，手机状态栏位于手机屏幕的最上方，那里一般显示了手机当前的网络状态、电池状态、事件等。Notification 不会打断用户当前的操作，支持异步的单击事件响应，程序一般由 NotificationManager 来管理，Notification Manager 负责发通知、清除通知等。它是一个系统 Service，必须通过 getSystemService()方法来获取。创建 Notification 的一般步骤如下。

（1）得到 NotificationManager，通过 getSystemService 方法得到 NotificationManager。

（2）构造一个 Notification 对象。

（3）设置 Notification 的属性参数。

（4）通过 NotificationManager 发送一个 Notification。

在界面上放置一个命令按钮，单击命令按钮时创建一个 Notification 的核心代码，如下

所示。

```
//得到 NotificationManager
NotificationManager  notificationManager = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);
 //实例化一个的 notification,并在实例化时设置图标、文本内容、发送时间
Notification notification = new Notification(R.drawable.image1, "notice",
System.currentTimeMillis());
//创建一个启动其他 Activity 的 Intent
Intent intent = new Intent(ThisActivity.this,OtherActivity.class);
//创建一个延迟发送的 Intent
PendingIntent pendingIntent
= PendingIntent.getActivity(getApplicationContext(), 0, intent, 1);
//设置事件信息
notification.setLatestEventInfo(getApplicationContext(),"title","a message",
pendingIntent);
//发送通知
 notificationManager.notify(NOTIFICATION_ID, notification);
```

如果想要取消 一个 Notification，只需要使用 NotificationManager 的 cancel 方法取消该 Notification 即可。

```
NotificationManager notificationManager = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);
notificationManager.cancel(NOTIFICATION_ID);
```

4.5 本章小结

菜单和对话框在用户界面中使用非常频繁，本章主要介绍了 Android 中菜单组件的特性及使用方法，通过实例详细阐述了如何通过不同的方法创建选项菜单、如何监听事件。如何创建上下文菜单及其使用。对话框在和用户进行交互时可以提高用户可操作性，本章通过实例详细展示了提示消息 Toast 和 Notification 的使用。

第 5 章　Intent 和 ContentProvider

Android 应用主要由四种组件组成，分别为 Activity、Broadcast、Service、ContentProvider，在这些组件（ContentProvider 除外）之间的通讯中，主要是由 Intent 协助完成，它适合用于传递数据量小的场合。而系统的多个应用程序之间，有时需要进行数据的共享与交换，Android 提供了 Content Provider 机制。

5.1　Intent

Android 中提供了 Intent 机制来协助应用间的交互与通讯，它封装了 Android 应用程序需要启动某个组件的“意图”，它不仅可用于应用程序之间，也可用于应用程序内部的 Activity/Service 之间的交互。可以通过 Intent 启动另一个 Activity、启动 Service、发起广播 Broadcast 等，并可以通过它传递数据。因此，Intent 起着一个媒体中介的作用，专门提供组件互相调用的相关信息，实现调用者与被调用者之间的解耦。

5.1.1　Intent 属性

Intent 由组件名称、执行动作描述 Action、该动作关联数据的描述等几部分组成。下面阐述各属性所代表的含义及其作用。

1）Component

指定 Intent 的目标组件的类名称。如果 Component 属性有指定的话，将直接使用它指定的组件，指定了这个属性以后，Intent 的其他所有属性都是可选的。

2）Action

也就是要执行的动作。使用一个字符串对所将执行的动作的描述，为了方便引用，Intent 类中定义了一些标准的动作，用户也可以根据需要自行定义 Action。下面是一些常用的 Action。

- **ACTION_CALL**　拨打 Data 里用 URI 表示的电话号码。
- **ACTION_MAIN**　启动项目的初始界面。
- **ACTION_VIEW**　常和特定的数据和 URI 配合使用，用于将数据和网站等显示给用户。

```
Uri oneUri=Uri.parse(“http://www.zzuli.edu.cn”);//指定 Uri 为网址 Intent
aIntent=new Intent(Intent. ACTION VIEW, oneUri);
startAcivtiy(aIntent);
```

- **ACTION_DIAL**　用于描述给用户打电话的动作。

```
Intent aIntent=new Intent(Intent. ACTION_DIAL, Uri.parse(“tel:123456”));
startAcivtiy(aIntent);
```

- ❑ **ACTION_EDIT** 打开数据里指定数据所对应的编辑程序。
- ❑ **ACTION_DELETE** 删除指定的数据。
- ❑ **ACTION_BATTERY_LOW** 警告电池电量低。
- ❑ **ACTION_HEADSET_PLUG** 耳机插入/拔掉设备。
- ❑ **ACTION_TIME_CHANGED** 系统时间已经改变。

3）Data

动作要操作的数据，Android 中采用指向数据的一个 Uri 来表示，Data 主要完成对 Intent 消息中数据的封装，不同类型的 Action 会有不同的 Data 封装。如 ACTION_EDIT 指定 Data 为文件 Uri，打电话为 tel: Uri，访问网络为 http: Uri，而由 Content Provider 提供的数据则为 content: URIs。

4）Category

它是对目标组件类别信息的描述。一个 Intent 对象可以包含多个 Category。Intent 类定义了许多 Category 常数，来表示 Intent 的不同类别，如下所示。

- ❑ **CATEGORY_DEFAULT** 表示默认的 Category。
- ❑ **CATEGORY_HOME** 设置该 Activity 随系统启动而运行。
- ❑ **CATEGORY_LAUNCHER** 表示该 Activity 是应用程序中最先被执行的 Activity。
- ❑ **CATEGORY_BROWSABLE** 该 Activity 能被浏览器安全调用。
- ❑ **CATEGORY_TAB** 表示目标 Activity 是可以嵌入到其 Activity 中的。
- ❑ **CATEGORY_PREFERENCE** 该 Activity 是参数面板。

5）Extras

Extras 中封装了一些额外的以键值对形式存在的附加信息。使用 Extras 可以为组件提供扩展信息，比如，如果要执行“发送电子邮件”这个动作，可以将电子邮件的标题、正文等保存在 Extras 里，传给电子邮件发送组件。Intent 可以通过 putExtras()与 getExtras()方法来存储和获取 Extras。

5.1.2 Intent Filter

每个过滤器描述组件的一种能力，即乐意接收的一组 Intent。实际上，它筛掉不想要的 Intents。一个 Intent 过滤器是一个 IntentFilter 类的实例。因为 Android 系统在启动一个组件之前必须知道它的能力，但是 Intent 过滤器通常不在 Java 代码中设置，而是在应用程序的清单文件 AndroidManifest.xml 中以<intent-filter>元素设置。但有一个例外，广播接收者的过滤器通过调用 Context.registerReceiver()动态地注册，它直接创建一个 IntentFilter 对象。一个过滤器有对应于 Intent 对象的动作、数据、种类的字段。

1）检测 Action

Action 主要的内容有 MAIN、VIEW、PICK、EDIT 等。清单文件中的<intent-filter>元素以<action>子元素列出动作，例如，

```
<intent-filter . . . >
      <action android:name="com.example.project.SHOW_CURRENT" />
      <action android:name="com.example.project.SHOW_RECENT" />
```




```
        <action android:name="com.example.project.SHOW PENDING" />
    …
</intent-filter>
```

一个过滤器必须至少包含一个<action>子元素，否则它将阻塞所有的 Intents。一般一个 Intent 只能设置一个 Action，如果 Intent 对象没有指定动作，将自动通过检查。但是一个 Intent-filter 可以设置多个 Action 过滤，只要一个满足即可完成 Action 验证。

2）检测 category

Intent-filter 同样可以设置多个 Category。当 Intent 中的 Category 与 Intent-filter 中的一个 Category 完全匹配时，便通过 Category 的检测，而其他的 Category 并不受影响。但是当 Intent-filter 没有设置 Category 时，只能与没有设置 Category 的 Intent 相匹配，原则上应该总是通过种类测试，而不管过滤器中有什么种类。但是有个例外，Android 对待所有传递给 Context.startActivity() 的隐式 Intent 好像它们至少包含"android.intent.category.DEFAULT"（对应 CATEGORY_DEFAULT 常量）。因此，活动想要接收隐式 Intent，必须要在 Intent 过滤器中包含"android.intent.category.DEFAULT"。

清单文件中的<intent-filter>元素以<category>子元素列出种类，例如，

```
<intent-filter …>
      <category android:name="android.intent.category.DEFAULT" />
      <category android:name="android.intent.category.BROWSABLE" />
  …
</intent-filter>
```

因此，对于一个 Intent 要通过种类检测，Intent 对象中的每个种类必须匹配过滤器中的一个，即过滤器能够列出额外的种类，但是 Intent 对象中的种类都必须能够在过滤器中找到，只有一个种类在过滤器列表中没有，就算种类检测失败。



"android.intent.action.MAIN"和"android.intent.category.LAUNCHER"设置，它们分别标记活动开始新的任务和带到启动列表界面。它们可以包含"android.intent.category. DEFAULT"到种类列表，也可以不包含。

3）检测 Data

类似的，清单文件中的<intent-filter>元素以<data>子元素列出数据，例如，

```
<intent-filter …>
    <data android:mimeType="video/mpeg" android:scheme="http" …/>
  <data android:mimeType="audio/mpeg" android:scheme="http" … />
  …
</intent-filter>
```

每个<data>元素指定一个 Uri 和数据类型（MIME 类型）。一个 Uri 由 scheme、host、port、path 四个部分组成，形式如下：scheme://host:port/path。

例如，下面的 Uri：

```
content://com.example.project:200/folder/subfolder/etc
```

scheme 是 content，host 是 com.example.project，port 是 200，path 是 folder/subfolder/etc。host 和 port 一起构成 Uri 的凭据（authority），如果 host 没有指定，port 也被忽略。 这四个



属性都是可选的，但它们之间并不都是完全独立的。要让 authority 有意义，scheme 必须也要指定。要让 path 有意义，scheme 和 authority 也都必须要指定。

<data>元素的 type 属性指定数据的 MIME 类型。Intent 对象和过滤器都可以用"*"通配符匹配子类型字段，如“text/*”，“audio/*”表示任何子类型。

数据检测既要检测 Uri，也要检测数据类型，规则如下。

- **一个 Intent 对象既不包含 Uri，也不包含数据类型**　仅当过滤器也不指定任何 Uri 和数据类型时，才不能通过检测，否则都能通过。
- **一个 Intent 对象包含 Uri，但不包含数据类型**　仅当过滤器也不指定数据类型，同时它们的 Uri 匹配，才能通过检测。例如，mailto:和 tel:都不指定实际数据。
- **一个 Intent 对象包含数据类型，但不包含 Uri**　仅当过滤也只包含数据类型且与 Intent 相同，才通过检测。
- **一个 Intent 对象既包含 Uri，也包含数据类型**　数据类型部分，只有与过滤器中之一匹配才算通过；Uri 部分，它的 Uri 要出现在过滤器中，或者它有 content:或 file: Uri，又或者过滤器没有指定 Uri。换句话说，如果它的过滤器仅列出了数据类型，组件假定支持 content:和 file:。

5.1.3　Intent 的解析

当应用程序发送一个 Inent 请求，系统会根据 Intent 的内容在注册的 IntentFilter 中选择适当的组件来响应。Intent 有两种基本使用方法：显式 Intent 和隐式 Intent。显式 Intent 是指在构造 Intent 对象时就指明该 Intent 的接收者是谁；隐式 Intent 是指发送者在构造 Intent 对象时，并没有指明接收者是谁，需要对它进行解析，才能知道该 Intent 的接收者。所以 Intent 的解析只针对隐式 Intent。对于隐式 Intent，Android 需要通过解析寻找处理该 Intent 的目标组件，如 Activity、Service 或 Broadcast Receiver。隐式 Intent 的使用有利于降低发送者和接收者之间的耦合。

Intent 解析机制主要是通过查找在 AndroidManifest.xml 文件中定义的 Intent 及其注册在该 Intent 上的所有的<intent -filter>，最终找到处理该 Intent 的组件。在整个解析过程中，主要通过 Intent 的 type、action、category 这三个方面来进行检查。若这三个方面的任何一个不匹配，Android 都不会将该隐式 Intent 传递给目标组件，其检测过程如下。

（1）动作检测

如果该 Intent 显式指明了 action，其目标组件的<intent-filter>的 action 列表中就必须包含在 Intent 中指明的 action，如果不包含就不能匹配。

（2）类别检测

如果 Intent 指定了一个或多个 category，则这些类别必须全部出现在组件的类别列表中。只有请求中所有的 category 与<intent-filter>的<category>列表完全匹配，该 Intent 才匹配成功。也就是说，Intent 中包含的所有类别必须包含在<intent-filter>的<category>的列表中。而<intent-filter>中多余的<category>并不会导致匹配失败。如果一个 IntentFilter 没有指定任何类别，则该 IntentFilter 只能匹配没有<category>的 Intent 请求。

（3）数据检测

如果该 Intent 没有指明 type 属性，系统将自动从该 Intent 的 Data 属性中获取数据类型，

其数据内容如果不是 Uri 格式指明的，将根据 Intent 中数据的 scheme 进行匹配，Intent 的 scheme 必须出现在目标组件的 scheme 列表中。也就是说其目标组件的<intent-filter>的 Data 列表中就必须包含在 Intent 中指明的数据类型，若不包含，则不能匹配。

5.1.4 Intent 的实现

通过 Intent 可以启动另一个 Activity、启动 Service、发起广播 Broadcast 等，并可以通过它传递数据。一种是显式调用，另一种是隐式调用。

1）显式调用

Intent 最常用的功能就是连接应用程序当中的各个 Activity。显式 Intent 直接用组件的名称定义目标组件，这种方式很直接。启动一个特定的 Activity 核心代码如下。

```
Intent intent = new Intent(源Activity名.this, 目标Activity名.class);
startActivity(intent);
```

或者首先创建 ComponentName 对象，并将该对象设置成 Intent 对象的 Component 属性，这样应用程序即可根据该 Intent 的“意图”去启动指定组件。

```
ComponentName comp = new ComponentName(源 Activity 名.this,目标 Activity
名.class);
Intent intent = new Intent();
intent.setComponent(comp);
startActivity(intent);
```

也可以启动不同工程项目的 Activity，前提条件是被调用的类已经安装在运行的模拟器或手机上，如果知道其包名和类名，可以采用如下方式进行调用：

```
Intent intent = new Intent();
intent.setClassName("com.example.test","com.example.test.OtherActivity");
startActivity(intent);
```

由于开发人员往往并不清楚别的应用程序的组件名称。因此，显式 Intent 更多用于在应用程序内部传递消息。

案例：在 SimpleIntentDemo 中通过监听命令按钮的单击动作跳到另外一屏。

（1）创建项目 SimpleIntentDemo，并创建 Activity 名为 SimpleIntentDemoActivity。

（2）打开 res/layout 创建名为 second.xml 的布局文件，其中只有一个 TextView 其提示信息为“这是第二屏”。

（3）打开 src 创建 SecondActivity，并重载 onCreate()方法，通过 setContentView (R.layout.second)设置其布局文件。

（4）为 SecondActivity 在 AndroidManifest.xml 中进行注册。

```
</activity>
  <activity android:name=".SecondActivity"
  android:label="@string/app name">
  /></activity>
```

（5）打开 res/layout/main.xml，对 main.xml 进行修改，添加一个提示信息为“跳转到



第二屏”的命令按钮，并设置其 id，android:id="@+id/button1"。

其中，SimpleIntentDemoActivity 代码如下。

```
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class SimpleIntentDemoActivity extends Activity {
   public void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.main);
     Button bt=  (Button)findViewById(R.id.button1);
     bt.setOnClickListener(new ClickLis());
   }
   class ClickLis implements OnClickListener{
              public void onClick(View v) {
                     Intent oneIntent=new Intent();
                      oneIntent.setClass(SimpleIntentDemoActivity.this,
                      SecondActivity.class);
                       startActivity(oneIntent);

            }
   }
}
```

2）隐式调用

Intent 机制更重要的作用在于其隐式的 Intent，即 Intent 的发送者不指定接收者，很可能不知道也不关心接收者是谁，而由 Android 框架去寻找最匹配的接收者。

❑ **最简单的隐式 Intent**

使用最简单的隐式调用不指定接收者，初始化 Intent 对象时，只是传入参数，用 Intent 调用系统中的组件，如设定 Action 为 Intent.ACTION_DIAL。

```
Intent intent = new Intent(Intent.ACTION DIAL);
startActivity(intent);
```

就会启动 Android 自带的打电话功能的 Dialer 程序。这里使用的构造函数的原型如下。

```
Intent(String action);
```

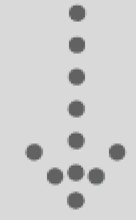
其中，Action 为 Intent 的常量，如 Intent.ACTION_DIAL、Intent.ACTION_SEND、Intent.ACTION_VIEW 等 Intent 的发送者只是指定了 Action。如果用户启动的 Activity 的描述信息正好与第三方 Activity 的描述信息相匹配，这个第三方的 Activity 就会被启动。

案例：在 SimpleImplicitIntent 中通过监听命令按钮的单击动作实现浏览指定的网页。

（1）创建项目 SimpleImplicitIntent，并创建 Activity 名为 SimpleImplicitIntentActivity。

（2）打开 res/layout 修改 main.xml 的布局文件，其中只有一个 Button 其提示信息为“打开网页”，并设置其 id，android:id="@+id/button"。

其中，SimpleImplicitIntentActivity 代码如下。



```
package com.chapt6;

import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class SimpleImplicitIntentActivity extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button bt = (Button) findViewById(R.id.button);
        bt.setOnClickListener(new OnClickListener() {
              public void onClick(View v) {
                    Uri myuri = Uri.parse("http://www.zzuli.edu.cn");
                    Intent intent = new Intent(Intent.ACTION VIEW, myuri);
                    startActivity(intent);
              }
        });
    }
}
```

在上述代码中修改 Intent 相关语句，可以通过 Intent 播放音频文件，相关语句如下。

```
Uri myuri=Uri.parse("file:///sdcard/song.mp3");
Intent intent = new Intent(Intent.ACTION VIEW);
intent.setDataAndType(myuri, "audio/mp3");
startActivity(intent);
```

❑ **增加接收者的隐式 Intent**

接收者如果希望能够接收某些 Intent，需要通过在 AndroidManifest.xml 中增加 Activity 的声明，并设置对应的 Intent Filter 和 Action，才能被 Android 的应用程序框架所匹配。

案例：通过监听命令按钮的单击动作，实现选择打开系统打电话程序和自定义打电话程序的选择，用户选择不同的按钮时跳到不同的界面，其实现效果如图 5-1 所示。

（1）创建项目 DialDemo，并创建 Activity 名为 MyDialActivity。

（2）打开 res/layout 修改 main.xml 的布局文件，其中只有一个 Button 其提示信息为“dialtel”，并设置其 id，android:id="@+id/button"

（3）打开 src 修改 MyDialActivity，并重载 onCreate() 方法，其代码如下。



图 5-1　具有接收者的隐式 Intent

```
package com.chapt6;

import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class MyDialActivity extends Activity {

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button bt = (Button) findViewById(R.id.button);
        bt.setOnClickListener(new OnClickListener() {

            public void onClick(View v) {
                Intent intent = new Intent(Intent.ACTION DIAL);
                startActivity(intent);

            }
        });
    }
}
```

（4）打开 src 新建一个 Activity:DialTelActivity，并存放在 com.chapt6 包中。

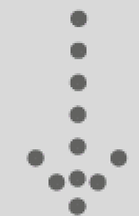
（5）打开 res/values/string.xml 文件，其内容如下。

```
<resources>
   <string name="hello">Hello World, MyDialActivity!</string>
   <string name="app name">DialDemo</string>
   <string name="dial">dialtel</string>
    <string name="title activity my dial">打开自定的拨号界面</string>
</resources>
```

（6）修改 AndroidManifest.xml 文件，将 DialTelActivity 的声明部分改为如下。

```
<activity android:name="com.chapt6.DialTelActivity"
    android:label="@string/title activity my dial">
    <intent-filter>
        <action android:name="android.intent.action.DIAL" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

针对 Intent.ACTION_DIAL，Android 框架找到了两个符合条件的 Activity，因此，它将这两个 Activity 分别列出，供用户选择。当选择“phone”，打开系统的拨打电话的界面，当选择“打开自定的拨号界面”，就会打开用户自定义的拨打电话界面，这里没有设置其布

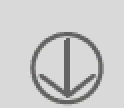


局，仅仅为空界面。

回过头来看我们是怎么做到这一点的。我们仅仅在 AndroidManifest.xml 文件中增加了下面的两行。

```
<action android:name="android.intent.action.DIAL" />
<category android:name="android.intent.category.DEFAULT" />
```

这两行修改了原来的 Intent Filter，这样这个 Activity 才能够接收到用户发送的 Intent。Intent 发送者设定 Action 来说明将要进行的动作，而 Intent 的接收者在 AndroidManifest.xml 文件中通过设定 Intent Filter 来声明自己能接收哪些 Intent。



5.1.5 Intent 中传递数据

Intent 除了定位目标组件外，另外一个职责就是传递数据信息。Intent 之间传递数据一般有两种常用的方法：一种是通过 data 属性；另一种是通过 extra 属性。data 属性是一种 URI，它可以指向 HTTP、FTP 等网络地址，也可以指向 ContentProvider 提供的资源。通过调用 Intent 的 setData 方法放入数据，使用 getData 方法取出数据。

如果需要启动 Android 内置的浏览器，使用下面的代码可将网址通过 data 属性传递给它。

```
Intent intent = new Intent(Intent.ACTION VIEW);
intent.setData(Uri.parse("http://www.google.com"));
startActivity(intent);
```

如果需要传递一下数据对象，则需要使用 extra 属性。Intent 提供了多个重载的方法来“携带”额外的数据，如下所示。

- ❑ **putExtra(String name, Xxx value)**　向 Intent 中放入 Xxx 类型的数据。
- ❑ **putIntegerArrayListExtra(String name, ArrayList<Integer> value)**　向 Intent 中放入 ArrayList 数据。
- ❑ **putStringArrayListExtra(String name, ArrayList<String> value)**　向 Intent 中放入 ArrayList 数据。
- ❑ **putExtras(Bundle extras)**　向 Intent 中通过 Bundle 对象传递数据。

如何获取传递的数据呢？可以使用 getIntent()方法得到上个 Activity 专递过来的 intent 内容。然后，根据数据的类型使用 intent 的 getXxxExtra(String key)方法获取相应的数据。

利用 Bundle 是一种比较方便的方法。Android 中的 Bundle 是一种类似于哈希表的数据结构，是一种键值对。可以将各种基本类型的数据保存在 Bundle 类中打包传输。在 Bundle 中定义了一种方法。

- ❑ **putXxx(String key, Xxx value)**　向 Bundle 中放入 int、long 等各种类型的数据。

为了取出 Intent 中携带的数据，Intent 中提供了如下方法。

- ❑ **getExtras()**　获取一个 Bundle 对象，然后使用 Bundle 的 get 方法来获取数据的值。
- ❑ **getXxx(String key)**　从 Bundle 中取出 Xxx 类型的数据。

- **getXxx(String key, Xxx defaultValue)** 从 Bundle 中取出 Xxx 类型的数据，如果取不到则使用 defaultValue。

案例：创建用户登录界面，让用户输入用户名和密码，当单击登录命令按钮时，跳转到另一个界面，显示欢迎信息，并把登录信息显示到当前界面上。其实现效果如图 5-2 所示。



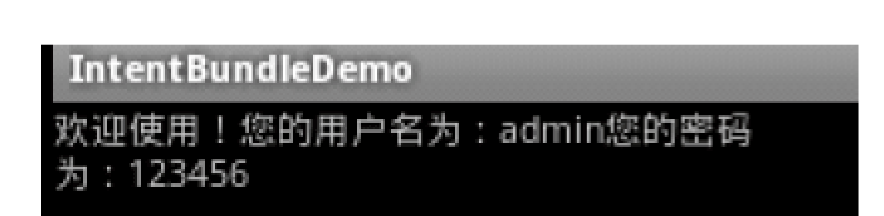


图 5-2　具有接收者的隐式 Intent

（1）创建项目 IntentBundleActivity，并创建 Activity 名为 IntentBundleActivity。

（2）打开 res/values/values/string.xml 文件，其内容如下。

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">IntentBundleDemo</string>
    <string name="username">用户名</string>
    <string name="password">密码</string>
    <string name="login">登录</string>
    <string name="cancel">取消</string>
</resources>
```

（3）打开 res/layout 修改 main.xml 的布局文件，其内容如下。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <LinearLayout
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:gravity="center">
     <TextView
        android:layout_width="0dip"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="@string/username" />
    <EditText
        android:id="@+id/username"
        android:layout_width="0dip"
        android:layout_height="wrap_content"
        android:layout_weight="1" />
    </LinearLayout>
    <LinearLayout
     android:layout_width="fill_parent"
```

```
        android:layout height="wrap content"
        android:gravity="center">
        <TextView
            android:layout width="0dip"
            android:layout height="wrap content"
            android:layout weight="1"
            android:text="@string/password" />
        <EditText
            android:id="@+id/userpassword"
            android:layout width="0dip"
            android:layout height="wrap content"
            android:layout weight="1" />
    </LinearLayout>
    <LinearLayout
     android:layout width="fill parent"
        android:layout height="wrap content"
        android:gravity="center">
        <Button
          android:layout width="wrap content"
              android:layout height="wrap content"
              android:id="@+id/buttonlogin"
              android:text="@string/login" />
        <Button
          android:layout width="wrap content"
              android:layout height="wrap content"
              android:id="@+id/buttoncancel"
              android:text="@string/cancel" />
    </LinearLayout>
</LinearLayout>
```

（4）打开 src 修改 IntentBundleActivity，并重载 onCreate()方法，其代码如下。

```
package com.chapt6;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;

public class IntentBundleActivity extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button bt1 = (Button) findViewById(R.id.buttonlogin);
        bt1.setOnClickListener(new OnClickListener() {

            public void onClick(View v) {
                 Bundle data = new Bundle();
```

```
                    //向 Bundle 中绑定数据，以键值对的形式
                    TextView username = (TextView) findViewById(R.id.username);
                  TextView userpassword = (TextView) findViewById(R.id.userpassword);
                   data.putString("name", username.getText().toString());
                   data.putString("password", userpassword.getText().toString());
                   Intent intent = new Intent(IntentBundleActivity.this,
                                      ShowResultActivity.class);
                   intent.putExtras(data);//把 Bundle 绑定到 Intent 中
                 startActivity(intent);
                }
            });
        }
    }
```

（5）打开 rec/layout 新建布局文件 showresultlayout.xml，其内容如下。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout width="match parent"
  android:layout height="match parent">
 <TextView
    android:layout_width="match_parent"
   android:layout height="match parent"
   android:id="@+id/showresult"
 ></TextView>
</LinearLayout>
```

（6）打开 src 定义一个 Activity：ShowResultActivity，并重载 onCreate()方法，其代码如下。

```
package com.chapt6;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;

public class ShowResultActivity extends Activity {
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.showresultlayout);
        TextView show=(TextView)findViewById(R.id.showresult);

        Intent intent=getIntent();
        //获取 Intent 中绑定的 Bundle
        Bundle result=intent.getExtras();
        String username=result.getString("name");
```

```
            String userpassword=result.getString("password");
            show.setText("欢迎使用!您的用户名为:"+username+"您的密码为:"+userpassword);
        }
    }
```

（7）为 ShowResultActivity 在 AndroidManifest.xml 中进行注册。

```
<activity
     android:name=".ShowResultActivity"
      android:label="@string/app name"
     ></activity>
```

5.1.6　在 Intent 中传递复杂对象

Android 的 Intent 之间传递对象有两种方法，一种是 Bundle.putSerializable(Key,Object);另一种是 Bundle.putParcelable(Key,Object)。方法中的 Object 要满足一定的条件，前者实现了 Serializable 接口，而后者实现了 Parcelable 接口。

Android 设计团队认为 Java 中的序列化太慢，难以满足 Android 的进程间通信需求，所以他们构建了 Parcelable 解决方案。Parcelable 要求显式地序列化类的成员，但最终序列化对象的速度将快很多。在 Android 运行环境中推荐使用 Parcelable 接口，它不但可以利用 Intent 传递，还可以在远程方法调用中使用。

实现 Parcelable 接口需要实现三个方法。

- **writeToParcel (Parcel dest, int flags)方法**　该方法将类的数据写入外部提供的 Parcel 中。
- **describeContents 方法**　返回内容描述信息的资源 ID，直接返回 0 就可以。
- **静态的 Parcelable.Creator<T>接口**　本接口有如下两个方法。
  - **createFromParcel(Parcel in)**　实现从 in 中创建出类的实例的功能。
  - **newArray(int size)**　创建一个类型为 T，长度为 size 的数组，returnnew T[size]即可。

案例：使用 Serializable 和 Parcelable 接口传递对象。

（1）创建项目 IntentObjectDemo，并包含一个 Activity：IntentObjectActivity。

（2）打开 src 创建类 SerializableUser，并实现 Serializable 接口。

```
package com.chapt6;
import java.io.Serializable;
public class SerializableUser implements Serializable {
    private String userName;
    private String passWord;
    public String getUserName() {
        return userName;
    }
    public void setUserName(String userName) {
        this.userName = userName;
    }
    public String getPassWord() {
        return passWord;
```

```
    }
    public void setPassWord(String passWord) {
        this.passWord = passWord;
    }
    public SerializableUser(String userName, String passWord) {
        this.userName = userName;
        this.passWord = passWord;
    }
}
```

（3）打开 src 创建类 ParcelableUser，并实现 Parcelable 接口。

```
package com.chapt6;
import android.os.Parcel;
import android.os.Parcelable;

public class ParcelableUser implements Parcelable {
    private String userName;
    private String password;
    public ParcelableUser() {

    }
    public ParcelableUser(String userName, String password) {
        this.userName = userName;
        this.password = password;
    }
    public String getUserName() {
        return userName;
    }
    public void setUserName(String userName) {
        this.userName = userName;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
    public int describeContents() {

        return 0;
    }
    public void writeToParcel(Parcel p, int arg1) {
        p.writeString(userName);
        p.writeString(password);
    }

    public static final Parcelable.Creator<ParcelableUser> CREATOR=new
    Creator<ParcelableUser>(){
       public ParcelableUser createFromParcel(Parcel source) {
                    ParcelableUser parcelableUser = new ParcelableUser();
```

```
                parcelableUser.userName = source.readString();
                parcelableUser.password = source.readString();
                return parcelableUser;
            }
            public ParcelableUser[] newArray(int size) {
                return new ParcelableUser[size];
                }
            };
}
```



（4）打开 res/layout/main.xml 文件，其内容如下。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout width="fill parent"
  android:layout height="fill parent"
  >
 <Button
    android:id="@+id/button1"
    android:layout width="wrap content"
    android:layout height="wrap content"
    android:text="传递 Serializable 对象"
    android:onClick="sendData" />
 <Button
    android:id="@+id/button2"
    android:layout width="wrap content"
    android:layout height="wrap content"
    android:text="传递 Parcelable 对象"
    android:onClick="sendData" />
</LinearLayout>
```

（5）打开 src 修改 IntentObjectActivity，并重载 onCreate()方法，其代码如下。

```
package com.chapt6;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;

public class IntentObjectActivity extends Activity {
  public void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.main);
  }
  public void sendData(View view){
              switch(view.getId()){
              case R.id.button1:
                  SerializableUser sUser = new SerializableUser("Admin",
                  "123456");
                  Intent intent = new Intent(this,ReceiveObjectActivity.
                  class);
```

```
                Bundle bundle = new Bundle();
                bundle.putInt("type", 1);
                bundle.putSerializable("serial", sUser);
                intent.putExtras(bundle);
                startActivity(intent);
                break;
        case R.id.button2:
                ParcelableUser pUser = new ParcelableUser("User", "123456");
                Intent intent1 = new Intent(this,ReceiveObjectActivity.
                class);
                Bundle bundle1 = new Bundle();
                bundle1.putInt("type", 2);
                bundle1.putParcelable("parcel", pUser);
                intent1.putExtras(bundle1);
                startActivity(intent1);
                break;
        }
    }
}
```

（6）打开 rec/layout 新建布局文件 objectreceiver.xml，其内容如下。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout width="match parent"
  android:layout height="match parent">
 <TextView
     android:layout width="match parent"
    android:layout height="match parent"
    android:id="@+id/showresult"
  ></TextView>
</LinearLayout>
```

（7）打开 src 定义一个 Activity: ReceiveObjectActivity，并重载 onCreate()方法，其代码如下。

```
package com.chapt6;
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class ReceiveObjectActivity extends Activity {
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.objectreceiver);
        TextView tv = (TextView) findViewById(R.id.showresult);
        Bundle bundle = getIntent().getExtras();
        int type = bundle.getInt("type");
        if (type == 1) {
            SerializableUser serializableUser = (SerializableUser) getIntent()
```

```
                        .getSerializableExtra("serial");
                tv.setText(serializableUser.getUserName() + "\n"
                        + serializableUser.getPassWord());
            } else {
                ParcelableUser parcelableUser = (ParcelableUser) getIntent()
                        .getParcelableExtra("parcel");
                tv.setText(parcelableUser.getUserName() + "\n"
                        + parcelableUser.getPassword());
            }
        }
    }
```

（8）为 ReceiveObjectActivity 在 AndroidManifest.xml 中进行注册。

```
<activity
    android:name=".ReceiveObjectActivity"
     android:label="@string/app name"
    ></activity>
```

5.2 ContentProvider

系统的多个应用程序之间，有时需要进行数据的共享与交换，Intent 只适合用于传递数据量小的场合，对于大的数据文件交互明显不合适。Android 提供了 ContentProvider 实现数据共享。

5.2.1 ContentProvider 简介

ContentProvider 是一个抽象类，可以理解为一个特殊的存储数据的类型，它提供了一套标准的接口来获取和操作数据。Android 自身也提供了现成的 Content Provider:Contents

可以把数据封装到 ContentProvider 中，从而使这些数据可以被其他的应用程序所共享，搭建起了所有应用程序之间数据交换的桥梁。

当应用继承 ContentProvider 类，并重写该类用于提供数据和存储数据的方法，就可以向其他应用共享其数据。虽然使用其他方法也可以对外共享数据，但数据访问方式会因数据存储的方式而不同，如采用文件方式对外共享数据，需要进行文件操作读写数据；采用 sharedpreferences 共享数据，需要使用 sharedpreferences API 读写数据。而使用 ContentProvider 共享数据的好处是统一了数据访问方式。

ContentProvider 类实现了一组标准的方法接口，从而能够让其他的应用程序保存或读取此 ContentProvider 的各种数据类型。在程序内可以通过实现 ContentProvider 的抽象接口将自己的数据显示出来，外界通过这个统一的接口来实现数据的增删改查。

当应用需要通过 ContentProvider 对外共享数据时，第一步需要继承 ContentProvider 并重写下面方法：

```
public class UserProvider extends ContentProvider{
   public boolean onCreate()
   public Uri insert(Uri uri, ContentValues values)
   public int delete(Uri uri, String selection, String[] selectionArgs)
   public int update(Uri uri, ContentValues values, String selection, String[]
   selectionArgs)
   public Cursor query(Uri uri, String[] projection, String selection,
   String[] selectionArgs, String sortOrder)
   public String getType(Uri uri)
}
```

第二步需要在 AndroidManifest.xml 使用<provider>对该 ContentProvider 进行配置，为了能让其他应用找到该 ContentProvider，ContentProvider 采用了 authorities（主机名/域名）对它进行唯一标识，你可以把 ContentProvider 看作是一个网站，authorities 就是它的域名。

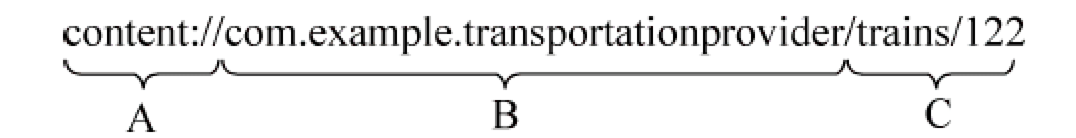
```
<provider android:name=".UserProvider"
    android:authorities="com.chapter.userprovider"/>
```

5.2.2 Uri、UriMatcher、ContentUris 和 ContentResolver 类简介

使用 ContentProvider，Uri 起到了关键作用，因为它决定了去访问哪个 ContentProvider。

1. Uri

Uri 代表了要操作的数据，Uri 主要包含了两部分信息：需要操作的 ContentProvider；对 ContentProvider 中的哪些数据进行操作，它由以下几部分组成。



content://com.provider.userprovider/userinfo/10

其中：

A 部分表示 ContentProvider 的 scheme，它已由 Android 所规定，内容为 content://。

B 部分表示主机名（或叫 Authority），是一个 ContentProvider 的唯一标识，外部调用者可以该标识来找到这个 ContentProvider。

C 部分表示路径（或叫 path），表示要操作的数据，在构建路径时应该根据业务而定。例如：

（1）要访问 userinfo 表中 ID 为 2 的记录，可构建的路径为：/userinfo/2；

（2）要操纵 userinfo 表中 ID 为 2 的记录的 username 字段，可以构建 userinfo/2/username 的路径；

（3）要访问 userinfo 表中的所有记录，可构建的路径为：/userinfo。

Uri 类中的 parse()方法，可以把一个字符串转换成 Uri，使用如下。

```
Uri uri = Uri.parse("content://com.chapt6.userprovider/userinfo");
```

在使用 Content Provider 时，几乎都会用到 Uri，如果是自定义的 Content Provider，通

常将 Uri 定义为常量，从而在简化开发的同时也提高程序的可维护性。

Uri 代表了要操作的数据，所以需要解析 Uri 来获取数据。Android 系统提供了 UriMatcher 和 ContentUris 来操作 Uri。

2. UriMatcher

该类用于匹配 Uri，其用法如下。

（1）首先把需要匹配 Uri 路径注册上。

```
//UriMatcher.NO MATCH 表示不匹配任何路径的返回码
UriMatcher oneMatcher = new UriMatcher(UriMatcher.NO MATCH);
//如果 match()方法匹配 content://com.chapt6.userprovider/userinfo 路径，返回匹配码 1。添加需要匹配 uri，如果匹配就会返回匹配码
oneMatcher.addURI("com.chapt6.userprovider ", " userinfo", 1);
//如果 match()方法匹配 content://com.chapt6.userprovider/userinfo/#路径，返回匹配码为 2，其中，#号表示通配符
```

oneMatcher.addURI("com.chapt6.userprovider ", " userinfo /#", 2);

（2）对 Uri 注册完后，通过 uriMatcher.match(uri)方法对 Uri 匹配，若匹配成功返回对应的匹配码。

oneMatcher.match(Uri.parse("content://com.chapt6.userprovider/userinfo/5"))，返回的匹配码为 2。

3. ContentUris

该类用于获取 Uri 路径后面的 ID 部分。其中，常用的两个方法 withAppendedId(uri, id) 和 parseId(uri)，分别用于为路径加上 ID 和获取 ID。

```
Uri uri =Uri.parse("content://com.chapt6.userprovider/userinfo/");
 Uri resultUri =ContentUris.withAppendedId(uri, 5);
该语句执行后生成 content://com.chapt6.userprovider/userinfo/5 的 uri
Uri uri =Uri.parse("content://com.provider.userprovider/userinfo/5");
 Long userid = ContentUris.parseId(uri);//获取的 id 为 5
```

4. ContentResolver

ContentResolver 是通过 ContentProvider 来获取与应用程序共享的数据，当外部应用需要对 ContentProvider 中的数据进行访问时，可以使用 ContentResolver 类来完成对数据的增、删、改和查询操作。

使用时首先通过 getContext().getContentResolver()获取 ContentResolver 对象，然后可以通过 ContentResolver 对象的 insert、delete、update、query 方法操作数据。

5.2.3 自定义 ContentProvider

下面通过一个示例展示如何自定义 Content Provider。

案例：通过 ContentProvider 来读取获取数据库里信息，并把结果通过 Logcat 进行显示输出。

（1）创建项目 DefineContentProvider，并包含一个 Activity:MainActivity。

（2）打开 src 创建类 DataBaseHelpe 继承 SQLiteOpenHelper。该类创建的数据库名为

userdb.db，表名为 userinfo。

```
package com.chapt6;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteDatabase.CursorFactory;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;

public class DataBaseHelper extends SQLiteOpenHelper {

    public static final String DB NAME = "userdb.db";
    public static final String TABLENAME = "userinfo";
    public static final int DB VERSION = 1;
    public static final String CREATETABLE = "create table " + TABLENAME
            + "( id integer primary key,username text, userpassword text);";

    public DataBaseHelper(Context context) {
        super(context, DB NAME, null, DB VERSION);
    }
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(CREATETABLE);
    }

    public void onUpgrade(SQLiteDatabase db,int oldVersion, int newVersion) {
        Log.i("Database update......", "Update database from " + oldVersion
                + " to " + newVersion);
        //删除旧的表
        db.execSQL("drop table if it exists " + TABLENAME);
        //创建新表
        onCreate(db);
    }
}
```

（3）打开 src 创建类 UserProvider 继承 ContentProvider，需要实现该类的六个方法。

```
package com.chapt6;

import android.content.ContentProvider;
import android.content.ContentUris;
import android.content.ContentValues;
import android.content.UriMatcher;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.net.Uri;

public class UserProvider extends ContentProvider {

    private DataBaseHelper dbh = null;
    //1.发布 Content Provider 的 Uri 地址
```

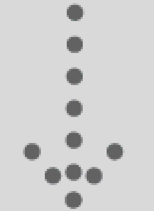
```
    private static final String AUTHORITY = "com.chapt6.userprovider";
    public static final Uri CONTENT URI = Uri
            .parse("content://com.chapt6.userprovider/userinfo");

    //2.注册需要匹配的 Uri
    private static UriMatcher uriMatcher = new UriMatcher(UriMatcher.NO MATCH);
    static {
        uriMatcher.addURI(AUTHORITY, "userinfo", 1);
        uriMatcher.addURI(AUTHORITY, "userinfo/#", 2);
    }

//该方法在 ContentProvider 创建后就会被调用，在其他应用第一次访问它时才会被创建
    public boolean onCreate() {
        //3.实例化 dbh
        dbh = new DataBaseHelper(getContext());
        return false;
    }

    //该方法用于返回当前 Uri 所代表数据的 MIME 类型
    public String getType(Uri uri) {
        //4.返回当前 Uri 所代表数据的 MIME 类型
        switch (uriMatcher.match(uri)) {
        case 1:
            return "vnd.android.cursor.dir/userinfo";
        case 2:
            return "vnd.android.cursor.item/userinfo";
        }
        return null;
    }

    //该方法用于供外部应用从 ContentProvider 删除数据。
    public int delete(Uri uri, String selection, String[] selectionArgs) {
        //5.实现删除方法
        SQLiteDatabase db = dbh.getWritableDatabase();
        int num = 0;//已经删除的记录数量
        switch (uriMatcher.match(uri)) {
        case 1:
            num = db.delete("userinfo", selection, selectionArgs);
            break;
        case 2:
            //获取 ID
            long id = ContentUris.parseId(uri);
            //在 selection 上增加条件 id=id
            if (selection == null) {
                selection = " id=" + id;
            } else {
                selection = " id=" + id + " and (" + selection + ")";
            }
            num = db.delete("userinfo", selection, selectionArgs);
            break;
        default:
```



```
            break;
        }
        //通知所有的观察者，数据集已经改变
        getContext().getContentResolver().notifyChange(uri, null);
        return num;
    }

    //该方法用于供外部应用往ContentProvider添加数据
    public Uri insert(Uri uri, ContentValues values) {
        //6.实现插入方法
        SQLiteDatabase db = dbh.getWritableDatabase();
        long id = db.insert("userinfo", null, values);
        if (id > -1) {//插入数据成功
            //构建新插入行的Uri
            Uri insertUri = ContentUris.withAppendedId(CONTENT_URI, id);
            //通知所有的观察者，数据集已经改变
            getContext().getContentResolver().notifyChange(insertUri, null);
        return insertUri;
        }
        return null;

    }
    //该方法用于供外部应用从ContentProvider中获取数据
    public Cursor query(Uri uri, String[] projection, String selection,
            String[] selectionArgs, String sortOrder) {
        SQLiteDatabase db = dbh.getReadableDatabase();
        Cursor cursor = null;
        switch (uriMatcher.match(uri)) {
        case 1://查询所有行
            cursor = db.query("userinfo",  //表名
                    null,                  //列的数组，null代表所有列
                    selection,             //where条件
                    selectionArgs,         //where条件的参数值的数组
                    null,                  //分组
                    null,                  //having
                    sortOrder);            //排序规则
            break;
        case 2://查询指定ID的行
                //获取ID
            long id = ContentUris.parseId(uri);
            //在selection上增加条件 id=id
            if (selection == null) {
                selection = " id=" + id;
            } else {
                selection = " id=" + id + " and (" + selection + ")";
            }
            cursor = db.query("userinfo",  //表名
                    null,                  //列的数组，null代表所有列
                    selection,             //where条件
                    selectionArgs,         //where条件的参数值的数组
                    null,                  //分组
```

```
                        null,       //having
                        sortOrder); //排序规则
                break;
            default:
                break;
            }
            return cursor;
        }

        //该方法用于供外部应用更新ContentProvider中的数据
        public int update(Uri uri, ContentValues values, String selection,
                String[] selectionArgs) {
            //7.实现修改方法
            SQLiteDatabase db = dbh.getWritableDatabase();
            int num = 0;//已经修改的记录数量
            switch (uriMatcher.match(uri)) {
            case 1:
                num = db.update("userinfo", values, selection, selectionArgs);
                break;
            case 2:
                //获取ID
                long id = ContentUris.parseId(uri);
                //在selection上增加条件_id=id
                if (selection == null) {
                    selection = " id=" + id;
                } else {
                    selection = " id=" + id + " and (" + selection + ")";
                }
                num = db.update("userinfo", values, selection, selectionArgs);
                break;
            default:
                break;
            }
            //通知所有的观察者，数据集已经改变
            getContext().getContentResolver().notifyChange(uri, null);
            return num;
        }
    }
```

说明：

如果操作的数据属于集合类型，那么，MIME 类型字符串应该以 vnd.android.cursor.dir/ 开头，如要得到 userinfo 所有记录的 Uri 为 content://com.chapt6.userprovider/userinfo，那么，返回的 MIME 类型字符串应该为“vnd.android.cursor.dir/userinfo”。

如果要操作的数据属于非集合类型数据，那么，MIME 类型字符串应该以 vnd.android.cursor.item/开头，如得到 id 为 2 的 userinfo 记录，Uri 为 content://com.chapt6.userprovider/userinfo/2，那么，返回的 MIME 类型字符串为“vnd.android.cursor.item/userinfo”。

（4）对 UserProvider 进行注册。在 AndroidManifest.xml 使用<provider>对该 ContentProvider 进行配置，代码如下。

```
<provider
    android:name="com.chapt6.UserProvider"
        android:authorities="com.chapt6.userprovider"
        android:exported="true"></provider>
```

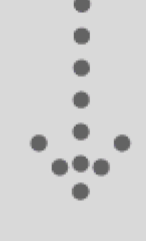
（5）对 UserProvider 进行访问。打开 src，在 MainActivity 的 onCreate 方法中增加代码，通过 UserProvider 进行数据的增、删、改、查的操作，并把结果通过 Logcat 显示输出，代码如下。

```
package com.chapt6;

import android.app.Activity;
import android.content.ContentResolver;
import android.content.ContentUris;
import android.content.ContentValues;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.util.Log;

public class MainActivity extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        ContentResolver cr = getContentResolver();
        //增加记录
        ContentValues values = new ContentValues();
        values.put("username", "admin");
        values.put("userpassword", "123456");
        cr.insert(UserProvider.CONTENT URI, values);
        values.clear();
        values.put("username", "zhangsan");
        values.put("userpassword", "666666");
        cr.insert(UserProvider.CONTENT URI, values);
        //查询所有记录
        Cursor cursor = cr.query(UserProvider.CONTENT URI, null, null, null,
                null);
        Log.i("after inserted", "------------------------------------");
        while (cursor.moveToNext()) {
            Log.i("after inserted", "id:" + cursor.getString(0) + " username:"
                + cursor.getString(1) +"userpassword:"+ cursor.getString(2));
        }
        cursor.close();
        //修改记录
        values.clear();
        values.put("username", "lisi");
        //构建的 Uri 为："content://com.chapt6.userprovider/userinfo/2"
        Uri uri = ContentUris.withAppendedId(UserProvider.CONTENT URI, 2);
        //修改 id 为 2 的记录
        cr.update(uri, values, null, null);
```



```
            //查询 id 为 2 的记录
            cursor = cr.query(uri, null, null, null, null);
            Log.i("after updated","------------------------------------");
            while (cursor.moveToNext()) {
                Log.i("after updated","id:" + cursor.getString(0) + "username:"
                + cursor.getString(1) + " userpassword:"+ cursor.getString(2));
            }
            cursor.close();
            //删除 id 为 2 的记录
            cr.delete(uri, null, null);
            //查询记录
            cursor = cr.query(UserProvider.CONTENT URI, null, null, null, null);
            Log.i("after deleted","------------------------------------");
            while (cursor.moveToNext()) {
                Log.i("after deleted","id:" + cursor.getString(0) + "username:"
                + cursor.getString(1) + " userpassword:"+ cursor.getString(2));
            }
            cursor.close();

        }
    }
```

5.2.4　系统 ContentProvider

Android 提供了一些主要数据类型的 ContentProvider，如音频、视频、图片和私人通讯录等。可在 android.provider 包下面找到一些 Android 提供的 ContentProvider。可以获得这些 ContentProvider，查询它们包含的数据，当然前提是已获得适当的读取权限。

案例：通过 ContentProvider 如何来读取短信信息，并把结果通过一个 TextView 进行显示。

Telephony Provider 提供了电话、短信和彩信相关数据的共享，可以通过该数据提供者访问手机中的短信信息，该数据库保存的路径为/data/data/com.android.providers.telephony/databases/mmssms.db，在 mmssms.db 数据库中，短信存储在 sms 表中，该表的结果如图 5-3 所示。

sms
_id : INTEGER
thread_id : INTEGER
address : TEXT
person : INTEGER
date : INTEGER
date_sent : INTEGER
protocol : INTEGER
read : INTEGER
status : INTEGER
type : INTEGER
reply_path_present : INTEGER
subject : TEXT
body : TEXT
service_center : TEXT
locked : INTEGER
error_code : INTEGER
seen : INTEGER

图 5-3　sms 表结构

其中，_id 表示该短信的 id；

thread_id 表示该短信所属的会话的 id，每个会话代表和一个联系人之间短信的群组；

address 表示该短信的发件人地址，手机号如+8613666666666；

person 表示该短信的发件人，返回一个数字就是联系人列表里的序号，陌生人为 null；

date 表示该短信的接收日期；

date_sent 表示该短信的发送日期；

protocol 协议，0 表示 SMS_RPOTO，1 表示 MMS_PROTO；

read 表示该短信是否已读；

type 表示该短信的类型，例如 1 表示接收类型，2 表示发送类型，3 表示草稿类型；

body 表示短信的内容。

通过查看 API 和源文件（\sources\android-19\android\provider\Telephony.java），发现主要的 Uri 如下。

```
content://sms/              所有短信
content://sms/inbox         收件箱
content://sms/sent          已发送
content://sms/draft         草稿
content://sms/outbox        发件箱
content://sms/failed        发送失败
content://sms/queued        待发送列表
```

实现步骤如下。

（1）创建项目 ReadSmsDemo，并包含一个 Activity:Main Activity。

（2）打开 src 目录修改 MainActivity.java 文件，增加短信读取代码。

```
package com.chapt6;

import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Locale;
import android.app.Activity;
import android.database.Cursor;
import android.database.sqlite.SQLiteException;
import android.net.Uri;
import android.os.Bundle;
import android.util.Log;
import android.widget.ScrollView;
import android.widget.TextView;

public class MainActivity extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        TextView tv = new TextView(this);
        tv.setText(getSmsInPhone());
        ScrollView sv = new ScrollView(this);
        sv.addView(tv);
        setContentView(sv);
```

```
        }
        public CharSequence getSmsInPhone() {
            final String SMS URI ALL = "content://sms/";
            StringBuilder smsBuilder = new StringBuilder();
            try {
                Uri uri = Uri.parse(SMS URI ALL);
                String[] projection = new String[] { " id", "address", "person",
                        "body", "date", "type" };
                Cursor cur = getContentResolver().query(uri, projection, null,
                        null, " date desc");
                if (cur.moveToFirst()) {
                    int index_Address = cur.getColumnIndex("address");
                    int index Person = cur.getColumnIndex("person");
                    int index Body = cur.getColumnIndex("body");
                    int index Date = cur.getColumnIndex("date");
                    int index Type = cur.getColumnIndex("type");

                    do {
                        String strAddress = cur.getString(index Address);
                        int intPerson = cur.getInt(index_Person);
                        String strbody = cur.getString(index Body);
                        long longDate = cur.getLong(index Date);
                        int intType = cur.getInt(index_Type);
                    SimpleDateFormat dateFormat = new SimpleDateFormat(
                                "yyyy-MM-dd hh:mm:ss", Locale.US);
                        Date d = new Date(longDate);
                        String strDate = dateFormat.format(d);
                        String strType = "";
                        if (intType == 1) {
                            strType = "接收";
                        } else if (intType == 2) {
                            strType = "发送";
                        } else {
                            strType = "null";
                        }
                        smsBuilder.append(strAddress + ", ");
                        smsBuilder.append(intPerson + ", ");
                        smsBuilder.append(strbody + ", ");
                        smsBuilder.append(strDate + ", ");
                        smsBuilder.append(strType);
                        smsBuilder.append("\n");
                    } while (cur.moveToNext());
                    if (!cur.isClosed()) {
                        cur.close();
                        cur = null;
                    }
                } else {
                    smsBuilder.append("没有短信!");
                } //end if
                smsBuilder.append("-------End!-------");
            } catch (SQLiteException ex) {
                Log.d("SQLiteException in getSmsInPhone", ex.getMessage());
```

```
            }
            return smsBuilder.toString();

        }
}
```

（3）修改 AndroidManifest.xml 文件，增加 android.permission.READ_SMS 权限。

```
<uses-permission android:name="android.permission.READ_SMS"/>
```

5.3 简单的通讯录管理程序

本程序实现简单的通讯录管理功能，主要实现添加联系人及对选定的联系人的信息进行编辑，可以浏览所有联系人的信息。主要对 Menu、Intent、ContentProvider、SQLite 数据库操作等进行综合练习。运行应用程序，按 Menu 菜单时出现 Add Contact 菜单，并显示已有的联系人信息，如图 5-4 所示。单击 AddContact，出现添加新联系人的界面，如图 5-5 所示，当单击 Cancel 按钮时，返回前一个界面。当单击 Save 按钮时显示添加后的信息联系人列表，并出现菜单，可以继续添加联系人，也可以对通讯录进行编辑，如图 5-6 所示。选定一个联系人，长按键可以把该联系人从通讯录中删除，如图 5-7 所示，删除后转向信息浏览界面。



图 5-4　运行开始界面



图 5-5　添加联系人界面



图 5-6　联系人信息浏览



图 5-7　删除选定的联系人

实现步骤如下。

（1）创建项目 SimpleContact，应用名 SimpleContact 并包含一个 Activity：ContactEditor。

（2）打开 rec/values/string.xml，添加对字符串的声明，其内容如下。

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app name">SimpleContacts</string>
    <string name="name">Name :</string>
    <string name="mobile">Mobile :</string>
    <string name="email">Email :</string>
    <string name="group">Group :</string>
    <string name="save"> Save </string>
    <string name="cancel"> Cancel </string>
    <string name="contact edit">Edit Contact</string>
    <string name="contact create">Create New Contact</string>
    <string name="error msg">error in SimpleContacts</string>
    <string name="menu revert">Revert</string>
    <string name="menu delete">Delete</string>
    <string name="menu discard">Discard</string>
    <string name="menu add">Add Contact</string>
    <string name="menu edit">Edit Contact</string>
</resources>
```

（3）打开 rec/laytout，创建布局文件 contact_editor.xml，其内容如下。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout width="fill parent"
    android:layout height="fill parent" android:background="@drawable/bg">

    <TableRow
      android:id="@+id/TableRow01"
      android:layout width="fill parent"
            android:layout height="wrap content">
            <TextView
              android:id="@+id/TextView01"
              android:layout width="wrap content"
                  android:layout height="wrap content"
                  android:text="@string/name"
                  android:textSize="16px"></TextView>
            <EditText
              android:id="@+id/EditText01"
              android:layout height="wrap content"
                  android:layout width="fill parent" />
    </TableRow>
    <TableRow
      android:id="@+id/TableRow02"
      android:layout width="fill parent"
         android:layout height="wrap content">
         <TextView
```

```
            android:id="@+id/TextView02"
            android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:textSize="16px"
                android:text="@string/mobile"></TextView>
          <EditText
            android:id="@+id/EditText02"
            android:layout_height="wrap_content"
            android:layout_width="fill_parent"></EditText>
      </TableRow>
      <TableRow
        android:id="@+id/TableRow03"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">
          <TextView
            android:id="@+id/TextView03"
            android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/email"
                android:textSize="16px"></TextView>
          <EditText
            android:id="@+id/EditText03"
            android:layout_height="wrap_content"
                android:layout_width="fill_parent"></EditText>
      </TableRow>
      <TableRow
        android:id="@+id/TableRow05"
        android:layout_width="fill_parent"
          android:layout_height="wrap_content">
          <Button
              android:id="@+id/Button01"
              android:layout_height="wrap_content"
                android:text="@string/save"
                android:textSize="16px"
                android:layout_width="wrap_content"></Button>
          <Button
            android:id="@+id/Button02"
            android:layout_height="wrap_content"
                android:text="@string/cancel"
                android:textSize="16px"
                android:layout_width="wrap_content"></Button>
      </TableRow>
</LinearLayout>
```

（4）打开 rec/laytout，创建用于显示查询信息的布局文件 contact_list.xml，其内容如下。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
```




```
  android:background="@drawable/bg">
<ListView
  android:id="@+id/ListView01"
  android:layout width="fill parent"
  android:layout height="wrap content"></ListView>
</LinearLayout>
```

（5）打开 rec/laytout，创建用于显示查询信息每一项的布局文件 contact_list_item.xml，其内容如下。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout width="fill parent"
  android:layout height="fill parent">
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
  android:id="@android:id/text1"
  android:layout width="fill parent"
  android:layout height="fill parent"
  android:textStyle="bold"
  android:textSize="18px"
  android:gravity="center vertical"
  android:paddingLeft="10px"
  android:singleLine="true"
/>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
  android:id="@android:id/text2"
  android:layout width="fill parent"
  android:layout height="fill parent"
  android:textStyle="normal"
  android:textSize="14px"
  android:gravity="center vertical"
  android:paddingLeft="10px"
  android:singleLine="true"
/>
</LinearLayout>
```

（6）打开 src，创建 DBHelper 类继承 SQLiteOpenHelper。

```
package com.chapt6.contact;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class DBHelper extends SQLiteOpenHelper {

    public static final String DATABASE NAME = "simplecontacts.db";
    public static final int DATABASE VERSION = 2;
    public static final String CONTACTS TABLE = "contacts";
    //创建数据库
```

```
        private static final String DATABASE CREATE = "CREATE TABLE " + CONTACTS
        TABLE +" ("
        + ContactColumn. ID+" integer primary key autoincrement,"
        + ContactColumn.NAME+" text,"+ ContactColumn.MOBILE+" text,"
        + ContactColumn.EMAIL+" text," + ContactColumn.CREATED+" long,"
        + ContactColumn.MODIFIED+" long);";

        public DBHelper(Context context) {
            super(context, DATABASE NAME, null, DATABASE VERSION);

        }

        public void onCreate(SQLiteDatabase db) {

            db.execSQL(DATABASE CREATE);
        }

        public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

            db.execSQL("DROP TABLE IF EXISTS " + CONTACTS TABLE);
            onCreate(db);
        }

    }
```

（7）其中（5）的类 ContactColumn 是为了方便使用，把数据表 contacts 的列名、列的索引值及其查询字段字符串在其中定义的类，打开 src，创建 ContactColumn 类并实现 BaseColumns 接口。

```
    package com.chapt6.contact;
    import android.provider.BaseColumns;
    public class ContactColumn implements BaseColumns {
        public ContactColumn() {
        }
        //列名
        public static final String NAME = "name";
        public static final String MOBILE = "mobileNumber";
        public static final String EMAIL = "email";
        public static final String CREATED = "createdDate";
        public static final String MODIFIED = "modifiedDate";
        //列 索引值
        public static final int  ID COLUMN = 0;
        public static final int NAME COLUMN = 1;
        public static final int MOBILE_COLUMN = 2;
        public static final int EMAIL COLUMN = 3;
        public static final int CREATED COLUMN = 4;
        public static final int MODIFIED COLUMN = 5;
        //查询结果
        public static final String[] PROJECTION = {  ID,//0
                NAME,//1
                MOBILE,//2
```

```
            EMAIL //3
    };
}
```

（8）打开 src 创建类 ContactsProvider 继承 ContentProvider，实现该类的六个方法。

```
package com.chapt6.contact;

import android.content.ContentProvider;
import android.content.ContentUris;
import android.content.ContentValues;
import android.content.UriMatcher;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteQueryBuilder;
import android.net.Uri;
import android.text.TextUtils;
import android.util.Log;

public class ContactsProvider extends ContentProvider {
    private static final String TAG = "ContactsProvider";
    private DBHelper dbHelper;
    private SQLiteDatabase contactsDB;
    public static final String AUTHORITY = "com.chapt6.provider.contact";
    public static final String CONTACTS_TABLE = "contacts";
    public static final Uri CONTENT_URI = Uri.parse("content://" + AUTHORITY
            + "/contacts");

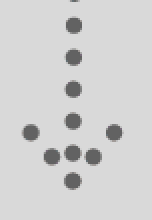
    public static final int CONTACTS = 1;
    public static final int CONTACT_ID = 2;
    private static final UriMatcher uriMatcher;
    static {
        uriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
        uriMatcher.addURI(AUTHORITY, "contacts", CONTACTS);
        //单独列
        uriMatcher.addURI(AUTHORITY, "contacts/#", CONTACT_ID);
    }

    public boolean onCreate() {
        dbHelper = new DBHelper(getContext());
        contactsDB = dbHelper.getWritableDatabase();
        return (contactsDB == null) ? false : true;
    }
    //删除指定数据列
    public int delete(Uri uri, String where, String[] selectionArgs) {
        //TODO Auto-generated method stub
        int count;
        switch (uriMatcher.match(uri)) {
        case CONTACTS:
            count = contactsDB.delete(CONTACTS_TABLE, where, selectionArgs);
```

```
            break;
        case CONTACT ID:
            String contactID = uri.getPathSegments().get(1);
            count = contactsDB.delete(CONTACTS TABLE,
                    ContactColumn. ID+ "=" + contactID
                    + (!TextUtils.isEmpty(where) ? " AND (" + where
                    + ")" : ""), selectionArgs);
            break;
        default:
            throw new IllegalArgumentException("Unsupported URI: " + uri);
        }
        getContext().getContentResolver().notifyChange(uri, null);
        return count;
    }

    //URI 类型转换
    public String getType(Uri uri) {
        switch (uriMatcher.match(uri)) {
        case CONTACTS:
            return "vnd.android.cursor.dir/contacts";
        case CONTACT ID:
            return "vnd.android.cursor.item/contacts";
        default:
            throw new IllegalArgumentException("Unsupported URI: " + uri);
        }
    }

    //插入数据
    public Uri insert(Uri uri, ContentValues initialValues) {
        if (uriMatcher.match(uri) != CONTACTS) {
            throw new IllegalArgumentException("Unknown URI " + uri);
        }
        ContentValues values;
        if (initialValues != null) {
            values = new ContentValues(initialValues);
            Log.e(TAG + "insert", "initialValues is not null");
        } else {
            values = new ContentValues();
        }
        Long now = Long.valueOf(System.currentTimeMillis());
        //设置默认值
        if (values.containsKey(ContactColumn.CREATED) == false) {
            values.put(ContactColumn.CREATED, now);
        }
        if (values.containsKey(ContactColumn.MODIFIED) == false) {
            values.put(ContactColumn.MODIFIED, now);
        }
        if (values.containsKey(ContactColumn.NAME) == false) {
            values.put(ContactColumn.NAME, "");
            Log.e(TAG + "insert", "NAME is null");
        }
```



```
        if (values.containsKey(ContactColumn.MOBILE) == false) {
            values.put(ContactColumn.MOBILE, "");
        }
        if (values.containsKey(ContactColumn.EMAIL) == false) {
            values.put(ContactColumn.EMAIL, "");
        }
        Log.e(TAG + "insert", values.toString());
        long rowId = contactsDB.insert(CONTACTS TABLE, null, values);
        if (rowId > 0) {
            Uri noteUri = ContentUris.withAppendedId(CONTENT URI, rowId);
            getContext().getContentResolver().notifyChange(noteUri, null);
            Log.e(TAG + "insert", noteUri.toString());
            return noteUri;
        }
        throw new SQLException("Failed to insert row into " + uri);
    }
    //查询数据
    public Cursor query(Uri uri, String[] projection, String selection,
                String[] selectionArgs, String sortOrder) {
        Log.e(TAG + ":query", " in Query");
        SQLiteQueryBuilder qb = new SQLiteQueryBuilder();
        qb.setTables(CONTACTS TABLE);
        switch (uriMatcher.match(uri)) {
        case CONTACT ID:
            qb.appendWhere(ContactColumn. ID + "="
                    + uri.getPathSegments().get(1));
            break;
        default:
            break;
        }
        String orderBy;
        if (TextUtils.isEmpty(sortOrder)) {
            orderBy = ContactColumn. ID;
        } else {
            orderBy = sortOrder;
        }
        Cursor c = qb.query(contactsDB, projection, selection, selectionArgs,
                null, null, orderBy);
        c.setNotificationUri(getContext().getContentResolver(), uri);
        return c;
    }
    //更新数据库
    public int update(Uri uri, ContentValues values, String where,
            String[] selectionArgs) {
        int count;
        Log.e(TAG + "update", values.toString());
        Log.e(TAG + "update", uri.toString());
        Log.e(TAG + "update :match", "" + uriMatcher.match(uri));
        switch (uriMatcher.match(uri)) {
        case CONTACTS:
            Log.e(TAG + "update", CONTACTS + "");
```

```
                count = contactsDB.update(CONTACTS TABLE, values, where,
                        selectionArgs);
                break;
            case CONTACT ID:
                String contactID = uri.getPathSegments().get(1);
                Log.e(TAG + "update", contactID + "");
            count = contactsDB.update(CONTACTS TABLE,values, ContactColumn. ID
                    + "="+ contacted + (!TextUtils.isEmpty(where) ? " AND
                (" + where+ ")" : ""), selectionArgs);
                break;
            default:
                throw new IllegalArgumentException("Unsupported URI: " + uri);
            }
            getContext().getContentResolver().notifyChange(uri, null);
            return count;
        }
    }
```

（9）打开 src，修改 ContactEditor。

```
package com.chapt6.contact;

import android.app.Activity;
import android.content.ContentValues;
import android.content.Intent;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;

public class ContactEditor extends Activity {

    private static final String TAG = "ContactEditor";

    private static final int STATE EDIT = 0;
    private static final int STATE INSERT = 1;

    private static final int REVERT ID = Menu.FIRST;
    private static final int DISCARD ID = Menu.FIRST + 1;
    private static final int DELETE ID = Menu.FIRST + 2;

    private int mState;
    private Uri mUri;
    private Cursor mCursor;
```





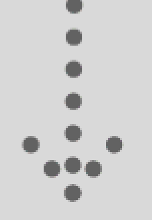

```
    private EditText nameText;
    private EditText mPhoneText;
    private EditText emailText;
    private Button saveButton;
    private Button cancelButton;

    private String originalNameText = "";
    private String originalMPhoneText = "";
    private String originalEmailText = "";

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        final Intent intent = getIntent();
        final String action = intent.getAction();
        Log.e(TAG + ":onCreate", action);
        if (Intent.ACTION_EDIT.equals(action)) {
            mState = STATE_EDIT;
            mUri = intent.getData();
        } else if (Intent.ACTION_INSERT.equals(action)) {
            mState = STATE_INSERT;
            mUri = getContentResolver().insert(intent.getData(), null);
            if (mUri == null) {
                Log.e(TAG + ":onCreate", "Failed to insert new Contact into "
                        + getIntent().getData());
                finish();
                return;
            }
            setResult(RESULT_OK, (new Intent()).setAction(mUri.toString()));

        } else {
            Log.e(TAG + ":onCreate", " unknown action");
            finish();
            return;
        }
        setContentView(R.layout.contact_editor);
        nameText = (EditText) findViewById(R.id.EditText01);
        mPhoneText = (EditText) findViewById(R.id.EditText02);
        emailText = (EditText) findViewById(R.id.EditText03);
        saveButton = (Button) findViewById(R.id.Button01);
        cancelButton = (Button) findViewById(R.id.Button02);
        saveButton.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                String text = nameText.getText().toString();
                if (text.length() == 0) {
                    setResult(RESULT_CANCELED);
                    deleteContact();
                    finish();
                } else {
                    updateContact();
                }
```




```
            }

        });
        cancelButton.setOnClickListener(new OnClickListener() {

            public void onClick(View v) {
                if (mState == STATE INSERT) {
                    setResult(RESULT CANCELED);
                    deleteContact();
                    finish();
                } else {
                    backupContact();
                }
            }
        });
        Log.e(TAG + ":onCreate", mUri.toString());
        //获得并保存原始联系人信息
        mCursor = managedQuery(mUri, ContactColumn.PROJECTION, null, null, null);
        mCursor.moveToFirst();
        originalNameText = mCursor.getString(ContactColumn.NAME COLUMN);
        originalMPhoneText = mCursor.getString(ContactColumn.MOBILE COLUMN);
        originalEmailText = mCursor.getString(ContactColumn.EMAIL COLUMN);
        Log.e(TAG, "end of onCreate()");
    }

    protected void onResume() {
        super.onResume();
        if (mCursor != null) {
            Log.e(TAG + ":onResume", "count:" + mCursor.getColumnCount());
            //读取并显示联系人信息
            mCursor.moveToFirst();
            if (mState == STATE EDIT) {
                setTitle(getText(R.string.contact edit));
            } else if (mState == STATE INSERT) {
                setTitle(getText(R.string.contact create));
            }
            String name = mCursor.getString(ContactColumn.NAME COLUMN);
            String mPhone = mCursor.getString(ContactColumn.MOBILE COLUMN);
            String email = mCursor.getString(ContactColumn.EMAIL COLUMN);
            nameText.setText(name);
            mPhoneText.setText(mPhone);
            emailText.setText(email);

        } else {
            setTitle(getText(R.string.error msg));
        }

    }
    protected void onPause() {
        super.onPause();
        if (mCursor != null) {
```

```
            String text = nameText.getText().toString();
            if (text.length() == 0) {
                Log.e(TAG + ":onPause", "nameText is null ");
                setResult(RESULT CANCELED);
                deleteContact();
                //更新信息
            } else {
        ContentValues values = new ContentValues();
        values.put(ContactColumn.NAME, nameText.getText().toString());
        values.put(ContactColumn.MOBILE, mPhoneText.getText().toString());
        values.put(ContactColumn.EMAIL, emailText.getText().toString());
        getContentResolver().update(mUri, values, null, null);
            }
        }
    }
    public boolean onCreateOptionsMenu(Menu menu) {
        if (mState == STATE EDIT) {
                menu.add(0,REVERT ID,1,R.string.menu revert).setShortcut
                ('0','r').setIcon(android.R.drawable.ic_menu_revert);
                menu.add(0,DELETE ID, 2, R.string.menu delete).setShortcut
                ('0','d').setIcon(android.R.drawable.ic menu delete);
        } else {
                menu.add(0,DISCARD ID,3,R.string.menu discard).setShortcut
               ('0','d').setIcon(android.R.drawable.ic menu delete);
        }
        return true;
    }
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
        case DELETE ID:
            deleteContact();
            finish();
            break;
        case DISCARD ID:
            cancelContact();
            break;
        case REVERT ID:
            backupContact();
            break;
        }
        return super.onOptionsItemSelected(item);
    }
    //删除联系人信息
    private void deleteContact() {
        if (mCursor != null) {
            mCursor.close();
            mCursor = null;
            getContentResolver().delete(mUri, null, null);
            nameText.setText("");
        }
```

```
        }
        //丢弃信息
        private void cancelContact() {
            if (mCursor != null) {
                deleteContact();
            }
            setResult(RESULT CANCELED);
            finish();
        }
        //更新 变更的信息
        private void updateContact() {
            if (mCursor != null) {
                mCursor.close();
                mCursor = null;
                ContentValues values = new ContentValues();
                values.put(ContactColumn.NAME, nameText.getText().toString());
                values.put(ContactColumn.MOBILE,mPhoneText.getText().toString());
                values.put(ContactColumn.EMAIL, emailText.getText().toString());
                getContentResolver().update(mUri, values, null, null);
            }
            setResult(RESULT CANCELED);
            finish();
        }

        //取消用，回退到最初的信息
        private void backupContact() {
            if (mCursor != null) {
                mCursor.close();
                mCursor = null;
                ContentValues values = new ContentValues();
                values.put(ContactColumn.NAME, this.originalNameText);
                values.put(ContactColumn.MOBILE, this.originalMPhoneText);
                values.put(ContactColumn.EMAIL, this.originalEmailText);
                getContentResolver().update(mUri, values, null, null);
            }
            setResult(RESULT CANCELED);
            finish();
        }
    }
```

（10）打开 src，创建 Contacts 类并继承 ListActivity，该类用于显示联系人的信息和编辑联系人界面。

```
    package com.chapt6.contact;
    import android.app.ListActivity;
    import android.content.ComponentName;
    import android.content.ContentUris;
    import android.content.Intent;
    import android.database.Cursor;
    import android.net.Uri;
    import android.os.Bundle;
```



```
import android.util.Log;
import android.view.ContextMenu;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ContextMenu.ContextMenuInfo;
import android.widget.AdapterView;
import android.widget.ListView;
import android.widget.SimpleCursorAdapter;

public class Contacts extends ListActivity {
        private static final String TAG = "Contacts";

        private static final int AddContact_ID = Menu.FIRST;
        private static final int EditContact_ID = Menu.FIRST+1;

    public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setDefaultKeyMode(DEFAULT_KEYS_SHORTCUT);
     Intent intent = getIntent();
    if (intent.getData() == null) {
        intent.setData(ContactsProvider.CONTENT_URI);
    }

    getListView().setOnCreateContextMenuListener(this);
    Cursor cursor = managedQuery(getIntent().getData(), ContactColumn.
    PROJECTION, null, null,null);
    //注册每个列表表示形式 : 姓名 + 手机号码
    SimpleCursorAdapter adapter = new SimpleCursorAdapter(this, R.layout.
    contact_list_item, cursor,
    new String[] { ContactColumn.NAME,ContactColumn.MOBILE }, new int[]
    { android.R.id.text1,android.R.id.text2 });
    setListAdapter(adapter);
    Log.e(TAG+"onCreate"," is ok");
 }

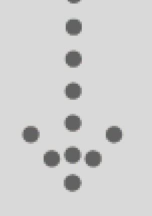
 public boolean onCreateOptionsMenu(Menu menu) {
    super.onCreateOptionsMenu(menu);
    menu.add(0, AddContact_ID, 0, R.string.menu_add).setShortcut('3', 'a')
        .setIcon(android.R.drawable.ic_menu_add);

    Intent intent = new Intent(null, getIntent().getData());
    intent.addCategory(Intent.CATEGORY_ALTERNATIVE);
    menu.addIntentOptions(Menu.CATEGORY_ALTERNATIVE,0,0,new ComponentName
                (this, Contacts.class), null, intent, 0, null);
    return true;
   }
 public boolean onPrepareOptionsMenu(Menu menu) {
    super.onPrepareOptionsMenu(menu);
    final boolean haveItems = getListAdapter().getCount() > 0;
    if (haveItems) {
```

```
        Uri uri = ContentUris.withAppendedId(getIntent().getData(),
        getSelectedItemId());
        Intent[] specifics = new Intent[1];
        pecifics[0] = new Intent(Intent.ACTION EDIT, uri);
        MenuItem[] items = new MenuItem[1];
        ntent intent = new Intent(null, uri);
        intent.addCategory(Intent.CATEGORY ALTERNATIVE);
        menu.addIntentOptions(Menu.CATEGORY ALTERNATIVE, 0, , null, specifics,
                intent, 0,items);
        if (items[0] != null) {
            items[0].setShortcut('1', 'e');
            }
        } else {
            menu.removeGroup(Menu.CATEGORY ALTERNATIVE);
        }
         return true;
    }

    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
        case AddContact ID:
            //添加 联系人
            startActivity(new Intent(Intent.ACTION INSERT, getIntent().
            getData()));
            return true;
        }
        return super.onOptionsItemSelected(item);
    }

  public void onCreateContextMenu(ContextMenu menu, View view, ContextMenuInfo
  menuInfo) {
    AdapterView.AdapterContextMenuInfo info;
    try {
        info = (AdapterView.AdapterContextMenuInfo) menuInfo;
    } catch (ClassCastException e) {
        return;
    }
    Cursor cursor = (Cursor) getListAdapter().getItem(info.position);
    if (cursor == null) {
        return;
    }
    menu.setHeaderTitle(cursor.getString(1));
    menu.add(0, EditContact ID, 0, R.string.menu delete);
 }
 public boolean onContextItemSelected(MenuItem item) {
    AdapterView.AdapterContextMenuInfo info;
    try {
        info = (AdapterView.AdapterContextMenuInfo) item.getMenuInfo();
    } catch (ClassCastException e) {
        return false;
    }
```



```
    switch (item.getItemId()) {
        case EditContact ID: {
            Uri noteUri = ContentUris.withAppendedId(getIntent().getData(),
            info.id);
            getContentResolver().delete(noteUri, null, null);
            return true;
        }
    }
    return false;
}

@Override
protected void onListItemClick(ListView l, View v, int position, long id) {
    Uri uri = ContentUris.withAppendedId(getIntent().getData(), id);

    String action = getIntent().getAction();
    if (Intent.ACTION_PICK.equals(action) || Intent.ACTION_GET_CONTENT.
    equals(action)) {

        setResult(RESULT OK, new Intent().setData(uri));
    } else {
        //编辑 联系人
        startActivity(new Intent(Intent.ACTION EDIT, uri));
    }
  }
}
```

（11）打开 AndroidManifest.xml，对定义的 provider、Activity 注册。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.chapt6.contact" android:versionCode="1"
    android:versionName="1.0.0">
    <application android:icon="@drawable/icon" android:label="@string/
    app name">

        <provider android:name="ContactsProvider"
            android:authorities="com.chapt6.provider.contact" />

        <activity android:name=".Contacts" android:label="@string/app name">
            <intent-filter>
                 <action android:name="android.intent.action.MAIN" />
                 <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
            <intent-filter>
                    <action android:name="android.intent.action.VIEW" />
                    <action android:name="android.intent.action.EDIT" />
                    <action android:name="android.intent.action.PICK" />
                    <category android:name="android.intent.category.DEFAULT" />
                    <data android:mimeType="vnd.android.cursor.dir/contacts" />
            </intent-filter>
```

```
            <intent-filter>
                    <action android:name="android.intent.action.GET CONTENT"/>
                    <category android:name="android.intent.category.DEFAULT"/>
                    <data android:mimeType="vnd.android.cursor.item/contacts"/>
            </intent-filter>
        </activity>
    <activity android:name=".ContactEditor" android:theme="@android:style/
    Theme.Light"
            android:label="ContactEditor">
            <intent-filter android:label="@string/menu edit">
                <action android:name="android.intent.action.VIEW" />
                <action android:name="android.intent.action.EDIT" />
            <action android:name="com.android.notepad.action.EDIT NOTE"/>
            <category android:name="android.intent.category.DEFAULT" />
            <data android:mimeType="vnd.android.cursor.item/contacts" />
            </intent-filter>

            <intent-filter>
                <action android:name="android.intent.action.INSERT" />
                <category android:name="android.intent.category.DEFAULT" />
            <data android:mimeType="vnd.android.cursor.dir/contacts" />
            </intent-filter>

        </activity>

    </application>
</manifest>
```

5.4　本章小结

本章主要介绍了 Android 系统中的 Intent 和 ContentProvider 的功能和用法，Android 使用 Intent 封装了应用程序的启动“意图”，讲述了 Intent 的各属性的功能和用法、如何在 Manifest.xml 文件中配置。ContentProvider 可以把应用程序的数据按照“固定规范”暴露出来，其他应用程序就可以通过其暴露出来的接口操作内部的数据。本章通过实例讲述 ContentProvider 组件的功能和用法，如何自定义 ContentProvider 和使用系统提供 ContentProvider，最后通过一个完整的实例通讯录管理系统把相关知识进行综合使用。

第 6 章　Android 下的多线程与事件处理机制

在应用程序设计时，如何给用户一个舒适惬意的体验感受一直是开发人员孜孜以求的最高目标。多线程机制的使用可以有效地加强应用程度处理状态与用户之间的互动，提升用户体验效果。另外，为了高效应对用户指令，Android 提供了强大的事件处理机制。本章介绍有关多线程与事件处理机制的相关知识。

6.1　Android 下的多线程

线程是进程中的一个实体，是被系统独立调度和分派的基本单位，线程本身不拥有系统资源，或只拥有少量运行中必不可少的资源，但它可与同属一个进程的其他线程共享进程中所拥有的全部资源。一个线程可以创建和撤销另一个线程，同一进程中的多个线程之间可以并发执行。线程之间相互制约，在运行中呈现间断性。线程有就绪、阻塞和运行三种基本状态。每一个程序都至少有一个线程，若程序只有一个线程，那就是程序本身。线程是程序中一个单一的顺序控制流程。在单个程序中同时运行多个线程完成不同的工作，称为多线程。

为什么要使用多线程呢？这是因为通过使用多线程可以提高资源的使用效率及系统的执行效率，主要体现在

- 使用线程可以合理分配“时间片”，把占用时间长的任务放置到后台处理；
- 实现更人性化的用户界面设计，如用户通过单击按钮触发了某些事件的处理，可以弹出一个进度条来显示处理的进度；
- 在一些需要等待的任务实现上，如用户输入、文件读写和网络收发数据等，利用多线程可以释放一些珍贵的资源（如内存占用等）。本章的第 1 节介绍多线程的的基本原理。

6.1.1　多线程机制的优缺点

Android 的多线程是基于 Linux 本身的多线程机制，而多线程之间的同步又是通过 Java 本身的线程同步。在 Android 应用使用多线程的主要优势体现在以下三个方面。

（1）避免 ANR，提升用户体验

在 Android 中，如果应用程序在某段时间内响应不够灵敏，系统会向用户显示一个对

话框，这个对话框称为应用程序无响应（ANR：Application Not Responding）对话框。用户可以选择“等待”而让程序继续运行，也可以选择“强制关闭”。默认情况下，在 Android 中 Activity 的最长执行时间是 5 秒，BroadcastReceiver 的最长执行时间则是 10 秒。一个流畅合理的应用程序中不应该出现 ANR，因为这样会带来不令人满意的用户体验，使用多线程可以避免 ANR。比如，在访问网络服务端时返回过慢、数据过多导致滑动屏幕不流畅或者 I/O 读取大资源时，则可以通过开启一个新线程来处理比较耗时的操作。这里需要提到我们开发时的一个事件处理的原则：把所有可能耗时的操作都放到其他线程去处理。参考代码如下。

```
new Thread() {
    public void run() {
     //耗时操作代码
    }
  }.start();
```

Android 中的 Main 线程在时间处理时若无法在 5 秒内得到响应，就会弹出 ANR 对话框。在 Main 线程中执行的方法如下。

- Activity 的生命周期方法，如 onCreate()、onStart()、onResume()等。
- 事件处理方法，如 onClick()、onItemClick()等。

一般来说，Activity 的 onCreate()、onStart()、onResume()方法的执行时间决定了应用首页打开的时间，要尽量把不必要的操作放到其他线程去处理，如果仍然很耗时，可以使用动态的 SplashScreen 告知用户应用正在运行。

（2）实现异步处理

当用户与应用交互时，事件处理方法的执行效率决定了应用程序的响应性能，分为以下两种情况。

同步，需要等待返回结果。如用户单击了“登录”按钮，需要等待服务端返回结果，那么，需要有一个进度条来提示用户程序的运行情况。

异步，不需要等待返回结果。异步的概念和同步相对。当一个异步过程调用发出后，调用者不能立刻得到结果。调用的部件在完成后，通过状态、通知和回调来通知调用者。例如，微博中的收藏功能，单击完“收藏”按钮后，会通知“收藏成功”而无需用户等待，这里就需要异步实现。

无论同步还是异步，事件处理都有可能比较耗时，此时需要放到其他线程中处理，处理完成后，再通知界面刷新。有一点需要注意，不是所有的界面刷新行为都需要放到 Main 线程中处理，例如，TextView 的 setText()方法需要在 Main 线程中，否则会抛出 CalledFromWrongThreadException 异常，而 ProgressBar 的 setProgress()方法则不需要在 Main 线程中处理。

（3）实现多任务

多任务是一个操作系统可以同时执行多个程序的能力。基本上，操作系统使用一个硬件时钟为同时运行的每个进程分配“时间片”。如果时间片足够小，并且机器负荷不重，那么在用户看来，所有的程序似乎在同时运行。程序使用多线程在后台执行长作业，从而用户仍然可以使用计算机进行其他工作。例如，用户向打印机发出打印命令，假如此时计算

机停止响应了，那岂不是用户就必须停止手上的工作来等待低速的打印机工作？所幸的是，用户在打印机工作的同时可以听音乐、画图、看电影等完成各种应用程序。这就是因为每一个程序被分成了独立不同的任务，使用多线程，即使某一部分任务失败了，也不会对其他的造成影响，不会导致整个程序崩溃。

总之，使用多线程可以获得更有效的 CPU 利用率、更好的系统可靠性、改善多处理器计算机的性能等，在许多应用中，可以同步调用资源。但纵使多线程具有以上诸多优势，切不可过多地使用多线程。这是因为过多使用多线程会出现数据同步的问题，需特别处理，在使用多线程时必须尽量保持每个线程的独立性不被其他线程干预。另外，如果一个程序有多个线程，那么，其他程序的线程必然只能占用更少的 CPU 时间，这样，还需要大量的 CPU 时间做线程调度，大量操作系统的内存空间维护每个线程的上下文信息，反而会降低系统的运行效率。

6.1.2 多线程的实现

Android 中的线程是基于 Java 定义的线程。一个应用程序中可能会包含多个线程（Thread），每个线程中都有一个 run()方法，run()方法内部的程序执行完毕后，所在的线程就自动结束。每个线程都有一个消息队列，用于不同的线程之间传递消息。

1）线程定义

Android 中定义线程的方法和 Java 是相同的，有两种方式：一种是 Thread；另一种是 Runnable。Thread 是一个类，而根据 Java 的继承要求，一个类只能有一个父类，所以继承了 Thread 的子类不能再继承其他类，限制了这种方法的使用。而 Runnable 是一个接口（interface），同样可以启动一个线程，不同的是它可以被多继承，参见代码如下。

```
package org.thread.demo;
class MyThread extends Thread{
 private String name;
 public MyThread(String name) {
  super();
  this.name = name;
 }
 public void run(){
  for(int i=0;i<10;i++){
   System.out.println("线程开始: "+this.name+",i="+i);
  }
 }
}
package org.thread.demo;
public class ThreadDemo01 {
 public static void main(String[] args) {
  MyThread mt1=new MyThread("线程 a");
  MyThread mt2=new MyThread("线程 b");
  mt1.run();
  mt2.run();
```

```
    }
  };
```

程序的运行很有规律，先执行第一个对象，然后执行第二个对象，并没有相互运行。在 jdk 的文档中可以发现，一旦调用 start()方法，则会通过 JVM 找到 run()方法。下面用 start()方法启动线程。

```
package org.thread.demo;
public class ThreadDemo01 {
 public static void main(String[] args) {
  MyThread mt1=new MyThread("线程 a");
  MyThread mt2=new MyThread("线程 b");
  mt1.start();
  mt2.start();
 }
};
```

这样程序可以正常完成交互式运行。那么，为什么要使用 start()方法启动多线程呢？这是因为在 JDK 的安装路径下，src.zip 是全部的 Java 源程序，通过此代码找到 Thread 中的 start()方法的定义，可以发现此方法中使用了 private native void start0();其中，native 关键字表示可以调用操作系统的底层函数，那么，这样的技术成为 JNI 技术（java Native Interface）

在实际开发中一个多线程的操作很少使用 Thread 类，而是通过 Runnable 接口完成。

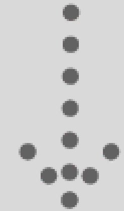
```
public interface Runnable{
 public void run();
}
```

例子：

```
package org.runnable.demo;
class MyThread implements Runnable{
 private String name;
 public MyThread(String name) {
  this.name = name;
 }
 public void run(){
  for(int i=0;i<100;i++){
   System.out.println("线程开始: "+this.name+",i="+i);
  }
 }
};
```

但是在使用 Runnable 定义的子类中没有 start()方法，只有 Thread 类中才有。此时观察 Thread 类，有一个构造方法：public Thread(Runnable targer)，此构造方法接受 Runnable 的子类实例，也就是说，可以通过 Thread 类来启动 Runnable 实现多线程。（start()可以协调系统的资源）

```
package org.runnable.demo;
import org.runnable.demo.MyThread;
public class ThreadDemo01 {
```



```
    public static void main(String[] args) {
     MyThread mt1=new MyThread("线程 a");
     MyThread mt2=new MyThread("线程 b");
     new Thread(mt1).start();
     new Thread(mt2).start();
    }
   };
```

在实际程序开发中多线程的实现多以 Runnable 接口为主，因为实现 Runnable 接口相比继承 Thread 类有如下好处：避免点继承的局限；一个类可以继承多个接口；适合于资源的共享等。

2）Handle、Message 和 Looper

Handler 主要接受子线程发送的数据，并用此数据配合主线程更新 UI。一个线程中只能有一个 Handler 对象，可以通过该对象向所在线程发送消息。Handler 主要有两种用途，一是实现一个定时任务，类似于 Windows 程序中的定时器功能，可以通过 Handler 对象向所在线程发送一个延时消息。等消息指定的时间到达后，通过 Handler 的消息处理函数完成指定的任务；二是在线程间传递数据。

❑ **完成定时任务**

在一个 Activity 内部实现定时器的功能，需要通过 Handler 对象的延迟发送消息方法来实现。Handler 有两类发送消息的方式：一类是 postXXX()方法，用于把一个 Runnable 对象发送到消息队列，从而当消息被处理时，能够执行 Runnable 对象；另一类是 sendXXX()方法，用于发送一个 Message 类型的消息到消息队列，当消息被处理时，系统会调用 Handler 对象定义的 handleMessage()方法处理该信息。

sendXXX()类包含以下方法。

- **sendEmptyMessage(int)**　空消息。
- **sendMessage(Message)**　发送 Message 指定的消息。
- **sendMessageAtTime(Message,long)**　在指定的时间点发送该消息。
- **sendMessageDelayed(Message,long)**　在指定的时间后发送该消息。

❑ **在线程之间传递数据**

如果一个进程获得了另一个进程的 handler，那么，这个进程就可以通过 handler.sendMessage(Message)方法向那个进程发送数据。基于这个机制，用户在处理多线程时可以新建一个 thread，这个 thread 拥有 UI 线程中的一个 handler。当 thread 处理完一些耗时的操作后通过传递过来的 handler 像 UI 线程发送数据，由 UI 线程去更新界面。

Thread 在默认的情况下，只要 run()函数执行完毕，线程就结束。但有时新建的线程需要接收消息并处理，因此，在新线程中，除了需要添加一个 Handler 对象外，还需要从线程的消息队列中取出消息，并负责分发消息，这就需要用 Looper 来实现了。事实上，Activity 内部就只有一个 Looper，只是 Activity 是一个特殊的 Thread，操作系统已经将其封装了而已。

在 Android 中 Handler 和 Message、Thread 有着很密切的关系。Handler 主要负责 Message 的分发和处理。Message 由一个消息队列进行管理，消息队列又是由一个 Looper 进行管理的。Android 系统中 Looper 负责管理线程的消息队列和消息循环。通过

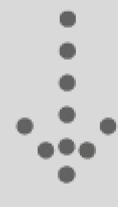
Loop.myLooper()可以得到当前线程的 Looper 对象，通过 Loop.getMainLooper()可以获得当前进程的主线程的 Looper 对象。 Android 系统的消息队列和消息循环都是针对具体线程的，一个线程可以存在（也可以不存在）一个消息队列和一个消息循环（Looper），特定线程的消息只能分发给本线程，不能进行跨线程，跨进程通讯。但是创建的工作线程默认是没有消息循环和消息队列的，如果想让该线程具有消息队列和消息循环，需要在线程中首先调用 Looper.prepare()来创建消息队列，然后调用 Looper.loop()进入消息循环。先来看一下 Looper.prepare()。

若现在线程中有一个 Looper 对象，其内部维护了一个消息队列 MQ。一个 Thread 只能有一个 Looper 对象，参照如下源码。

```
public class Looper {
    //每个线程中的 Looper 对象其实是一个 ThreadLocal，即线程本地存储(TLS)对象
    private static final ThreadLocal sThreadLocal = new ThreadLocal();
    //Looper 内的消息队列
    final MessageQueue mQueue;
    //当前线程
    Thread mThread;
    //其他属性
    //每个 Looper 对象中有它的消息队列，和它所属的线程
    private Looper() {
        mQueue = new MessageQueue();
        mRun = true;
        mThread = Thread.currentThread();
    }
    //我们调用该方法会在调用线程的 TLS 中创建 Looper 对象
    public static final void prepare() {
        if (sThreadLocal.get() != null) {
            //试图在有 Looper 的线程中再次创建 Looper 将抛出异常
            throw new RuntimeException("Only one Looper may be created per
            thread");
        }
        sThreadLocal.set(new Looper());
}
}
```

调用 loop 方法后，Looper 线程就开始真正工作了，它不断从自己的 MQ 中取出队头的消息（也叫任务）执行，其源码分析如下。

```
public static final void loop() {
        Looper me = myLooper();          //得到当前线程 Looper
        MessageQueue queue = me.mQueue; //得到当前 looper 的 MessageQueue
        Binder.clearCallingIdentity();
        final long ident = Binder.clearCallingIdentity();
        //开始循环
        while (true) {
            Message msg = queue.next(); //取出 message
            if (msg != null) {
                if (msg.target == null) {
```



```
                //message 没有 target 为结束信号，退出循环
                return;
            }
            //日志
            if (me.mLogging!= null) me.mLogging.println(
                    ">>>>> Dispatching to " + msg.target + " "
                    + msg.callback + ": " + msg.what
                    );
            msg.target.dispatchMessage(msg);
            //将处理工作交给 message 的 target，即 handler
            if (me.mLogging!= null) me.mLogging.println(
                    "<<<<< Finished to    " + msg.target + " "
                    + msg.callback);
            final long newIdent = Binder.clearCallingIdentity();
            if (ident != newIdent) {
                Log.wtf("Looper", "Thread identity changed from 0x"
                        + Long.toHexString(ident) + " to 0x"
                        + Long.toHexString(newIdent)+"while dispatching to"
                        + msg.target.getClass().getName() + " "
                        + msg.callback + " what=" + msg.what);
            }
            msg.recycle();  //回收 message 资源
        }
    }
}
```

除了 prepare()和 loop()方法，Looper 类还提供了如下方法。

Looper.myLooper()得到当前线程 looper 对象：getThread()得到 looper 对象所属线程：quit()方法结束 looper 循环等。

handler 扮演了往 MQ 上添加消息和处理消息的角色（只处理由自己发出的消息），即通知 MQ 它要执行一个任务（sendMessage），并在 loop 到自己的时候执行该任务（handleMessage），整个过程是异步的。handler 创建时会关联一个 looper，默认的构造方法将关联当前线程的 looper，不过这也是可以 set 的。LooperThread 类加入 Handler 的实现代码如下。

```
public class LooperThread extends Thread {
    private Handler handler1;
    private Handler handler2;
    @Override
    public void run() {
        //将当前线程初始化为 Looper 线程
        Looper.prepare();
        //实例化两个 handler
        handler1 = new Handler();
        handler2 = new Handler();
        //开始循环处理消息队列
        Looper.loop();
    }
}
```

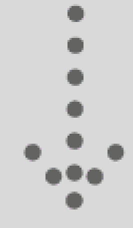
一个线程可以有多个 Handler，但是只能有一个 Looper。Handler 可以在任意线程发送消息，它首先创建消息，然后根据 Looper 找到相关联的消息队列，然后这些消息会被添加到关联的消息队列上。handler 是在它关联的 looper 线程中处理消息的。Looper 首先取出消息队列的头消息，对应的 Handler 执行 handlerMessage，最后返回 Looper 继续执行。

这就解决了 Android 不能在其他非主线程中更新 UI 的问题。Android 的主线程也是一个 looper 线程，用户在其中创建的 handler 默认将关联主线程消息队列。因此，利用 handler 的一个 solution 就是在 activity 中创建 handler 并将其引用传递给 worker thread，worker thread 执行完任务后使用 handler 发送消息通知 activity 更新 UI。

3）线程间的消息传递

虽说特定线程的消息只能分发给本线程，不能进行跨线程通讯，但是由于可以通过获得线程的 Looper 对象来进行曲线的实现不同线程间消息的传递，代码如下。

```
package com.mytest.handlertest;
import android.app.Activity;
import android.graphics.Color;
import android.os.Bundle;
import android.os.Handler;
import android.os.Looper;
import android.os.Message;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.ViewGroup.LayoutParams;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.TextView;
public class HandlerTest extends Activity implements OnClickListener{
    private String TAG = "HandlerTest";
    private boolean bpostRunnable = false;
    private NoLooperThread noLooperThread = null;
    private OwnLooperThread ownLooperThread = null;
    private ReceiveMessageThread receiveMessageThread =null;
    private Handler  mOtherThreadHandler=null;
    private EventHandler mHandler = null;
    private Button btn1 = null;
    private Button btn2 = null;
    private Button btn3 = null;
    private Button btn4 = null;
    private Button btn5 = null;
    private Button btn6 = null;
    private TextView tv = null;
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       LinearLayout layout = new LinearLayout(this);
       LinearLayout.LayoutParams params = new LinearLayout.LayoutParams
       (250, 50);
```



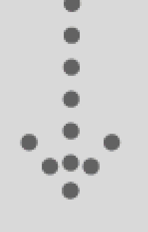
```
        layout.setOrientation(LinearLayout.VERTICAL);
        btn1 = new Button(this);
        btn1.setId(101);
        btn1.setText("message from main thread self");
        btn1.setOnClickListener(this);
        layout.addView(btn1, params);
        btn2 = new Button(this);
        btn2.setId(102);
        btn2.setText("message from other thread to main thread");
        btn2.setOnClickListener(this);
        layout.addView(btn2,params);
        btn3 = new Button(this);
        btn3.setId(103);
        btn3.setText("message to other thread from itself");
        btn3.setOnClickListener(this);
        layout.addView(btn3, params);
        btn4 = new Button(this);
        btn4.setId(104);
        btn4.setText("message with Runnable as callback from other thread to
        main thread");
        btn4.setOnClickListener(this);
        layout.addView(btn4, params);
        btn5 = new Button(this);
        btn5.setId(105);
        btn5.setText("main thread's message to other thread");
        btn5.setOnClickListener(this);
        layout.addView(btn5, params);
        btn6 = new Button(this);
        btn6.setId(106);
        btn6.setText("exit");
        btn6.setOnClickListener(this);
        layout.addView(btn6, params);
        tv = new TextView(this);
        tv.setTextColor(Color.WHITE);
        tv.setText("");
        params = new LinearLayout.LayoutParams(LayoutParams.FILL PARENT,
        LayoutParams.WRAP CONTENT);
        params.topMargin=10;
        layout.addView(tv, params);
        setContentView(layout);
        receiveMessageThread = new ReceiveMessageThread();
        receiveMessageThread.start();
    }

    class EventHandler extends Handler{
        public EventHandler(Looper looper){
            super(looper);
        }

        public EventHandler(){
            super();
```

```
        }
        @Override
        public void handleMessage(Message msg) {
            // TODO Auto-generated method stub
            super.handleMessage(msg);
            Log.e(TAG,"CurrentThread id:----------+>" + Thread.currentThread().
            getId());
            switch(msg.what){
            case 1:
                tv.setText((String)msg.obj);
                break;
            case 2:
                tv.setText((String)msg.obj);
                noLooperThread.stop();
                break;
            case 3:
                //不能在非主线程的线程里面更新 UI，所以这里通过 log 打印信息
                Log.e(TAG,(String)msg.obj);
                ownLooperThread.stop();
                break;
            default:
                Log.e(TAG,(String)msg.obj);
                break;
            }
        }
    }
     //ReceiveMessageThread has his own message queue by execute Looper.
     prepare();
    class ReceiveMessageThread extends Thread {
        @Override
        public void run(){
            Looper.prepare();
            mOtherThreadHandler= new Handler(){
                @Override
                public void handleMessage(Message msg) {
                    // TODO Auto-generated method stub
                    super.handleMessage(msg);
                    Log.e(TAG,"-------+>"+(String)msg.obj);
                    Log.e(TAG, "CurrentThread id:----------+>" + Thread.current
                    Thread().getId());
              }
             };
             Log.e(TAG, "ReceiveMessageThread id:--------+>" + this.getId());
             Looper.loop();
        }
    }

    class NoLooperThread extends Thread {
        private EventHandler mNoLooperThreadHandler;
        @Override
        public void run() {
```

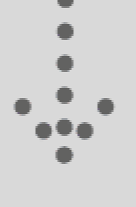


第 6 章　Android 下的多线程与事件处理机制

```
            Looper myLooper = Looper.myLooper();
            Looper mainLooper= Looper.getMainLooper();
            String msgobj;
            if(null == myLooper){
            //这里获得的是主线程的 Looper,由于 NoLooperThread 没有自己的 looper 所
            以这里肯定会被执行
                mNoLooperThreadHandler = new EventHandler(mainLooper);
                msgobj = "NoLooperThread has no looper and handleMessage
                function executed in main thread!";
            } else{
                mNoLooperThreadHandler = new EventHandler(myLooper);
                msgobj = "This is from NoLooperThread self and handleMessage
                function executed in NoLooperThread!";
            }
            mNoLooperThreadHandler.removeMessages(0);
            if(bpostRunnable == false){
            //send message to main thread
                Message msg = mNoLooperThreadHandler.obtainMessage(2, 1, 1,
                msgobj);
                mNoLooperThreadHandler.sendMessage(msg);
                Log.e(TAG, "NoLooperThread id:--------+>" + this.getId());
            }else{
                //下面 new 出来的实现了 Runnable 接口的对象中 run 函数是在 Main Thread
                中执行，不是在 NoLooperThread 中执行
                //注意 Runnable 是一个接口，它里面的 run 函数被执行时不会再新建一个线程
                mNoLooperThreadHandler.post(new Runnable(){
                    public void run() {
                        // TODO Auto-generated method stub
                        tv.setText("update UI through handler post runnalbe
                         mechanism!");
                        Log.e(TAG, "update UI id:--------+>" + Thread.current
                        Thread().getId());
                        noLooperThread.stop();
                    }
                }
            }
        }
    }
    class OwnLooperThread extends Thread{
        private EventHandler mOwnLooperThreadHandler = null;
        @Override
        public void run() {
            Looper.prepare();
            Looper myLooper = Looper.myLooper();
            Looper mainLooper= Looper.getMainLooper();
            String msgobj;
            if(null == myLooper){
                mOwnLooperThreadHandler = new EventHandler(mainLooper);
                msgobj = "OwnLooperThread has no looper and handleMessage
                function executed in main thread!";
```

```
            }
        else{
             mOwnLooperThreadHandler = new EventHandler(myLooper);
            msgobj = "This is from OwnLooperThread self and handleMessage
            function executed in NoLooperThread!";
            }

            mOwnLooperThreadHandler.removeMessages(0);
            //给自己发送消息
            Message msg = mOwnLooperThreadHandler.obtainMessage(3,1,1,msgobj);
            mOwnLooperThreadHandler.sendMessage(msg);
            Looper.loop();
        }
    }
    public void onClick(View v) {
        // TODO Auto-generated method stub
        switch(v.getId()){
        case 101:
            //主线程发送消息给自己
            Looper looper = Looper.myLooper();//get the Main looper related
            with the main thread
            //如果不给任何参数的话会用当前线程对应的Looper(这里就是Main Looper)为
            Handler里面的成员mLooper赋值
            mHandler = new EventHandler(looper);
            //清除整个MessageQueue里的消息
            mHandler.removeMessages(0);
            String obj = "This main thread's message and received by itself!";
            Message msg = mHandler.obtainMessage(1,1,1,obj);
            //将Message对象送入到main thread的MessageQueue里面
            mHandler.sendMessage(msg);
            break;
        case 102:
            //other线程发送消息给主线程
            bpostRunnable = false;
            noLooperThread = new NoLooperThread();
            noLooperThread.start();
            break;
        case 103:
             //other thread获取它自己发送的消息
            tv.setText("please look at the error level log for other thread
            received message");
            ownLooperThread = new OwnLooperThread();
            ownLooperThread.start();
            break;
        case 104:
            //other thread通过Post Runnable方式发送消息给主线程
            bpostRunnable = true;
            noLooperThread = new NoLooperThread();
            noLooperThread.start();
            break;
        case 105:
```



```
            //主线程发送消息给 other thread
            if(null!=mOtherThreadHandler){
            tv.setText("please look at the error level log for other thread
            received message from main thread");
            String msgObj = "message from mainThread";
            Message mainThreadMsg = mOtherThreadHandler.obtainMessage(1, 1, 1,
            msgObj);
            mOtherThreadHandler.sendMessage(mainThreadMsg);
            }
            break;
        case 106:
            finish();
            break;
        }
    }
}
```

6.2 事件处理机制

Android 提供了两套强大的事件处理机制：一个是基于监听的事件处理机制；另一个是基于回调的事件处理机制。对于 Android 基于监听的事件处理，主要的做法是为 Android 界面组件绑定特定的事件监听器。

对于 Android 基于回调的事件处理，主要的方法是重写 Android 组件特定的回调方法或重写 Activity 来完成的。

6.2.1 基于监听接口的事件处理

对于一个 Android 应用程序来说，事件处理是必不可少的，用户与应用程序之间的交互便是通过事件处理来完成的。关于 Android 事件处理模型应该注意以下几点。

（1）事件源与事件监听器。当用户与应用程序交互时，一定是通过触发某些事件来完成的，让事件来通知程序应该执行哪些操作，在这个繁杂的过程中主要涉及两个对象，事件源与事件监听器。

（2）事件源指的是事件所发生的控件，各控件在不同情况下触发的事件不尽相同，而且产生的事件对象也可能不同。

（3）监听器则是用来处理事件的对象，实现了特定的接口，根据事件的不同重写不同的事件处理方法来处理事件。

（4）将事件源与事件监听器联系到一起，就需要为事件源注册监听，当事件发生时，系统才会自动通知事件监听器来处理相应的事件。

事件处理的过程一般分为三步，如下所示。

（1）为事件源对象添加监听，这样当某个事件被触发时，系统才会知道通知谁来处理该事件。

（2）当事件发生时，系统会将事件封装成相应类型的事件对象，并发送给注册到事件

源的事件监听器。

（3）当监听器对象接收到事件对象之后，系统会调用监听器中相应的事件处理方法来处理事件并给出响应。

主要的监听器接口如下。

1）OnClickListener 接口

功能	该接口处理的是单击事件。在触控模式下，是在某个 View 上按下并抬起的组合动作，而在键盘模式下，是某个 View 获得焦点后单击确定键或按下轨迹球事件
回调方法	public void onClick(View v) 说明：参数 v 便为事件发生的事件源

2）OnLongClickListener 接口

功能	OnLongClickListener 接口与之前介绍的 OnClickListener 接口原理基本相同，只是该接口为 View 长按事件的捕捉接口，即当长时间按下某个 View 时触发的事件
回调方法	public boolean onLongClick(View v) 说明：参数 v 为事件源控件，当长时间按下此控件时才会触发该方法。 返回值：该方法的返回值为一个 boolean 类型的变量，当返回 true 时，表示已经完整地处理了这个事件，并不希望其他的回调方法再次进行处理；当返回 false 时，表示没有完全处理完该事件，更希望其他方法继续对其进行处理

3）OnFocusChangeListener 接口

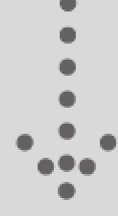
功能	OnFocusChangeListener 接口用来处理控件焦点发生改变的事件。如果注册了该接口，当某个控件失去焦点或获得焦点时都会触发该接口中的回调方法
回调方法	public void onFocusChange(View v, Boolean hasFocus) 说明：参数 v 为触发该事件的事件源； 参数 hasFocus 表示 v 的新状态，即 v 是否是获得焦点

4）OnKeyListener 接口

功能	OnKeyListener 是对手机键盘进行监听的接口，通过对某个 View 注册该监听，当 View 获得焦点并有键盘事件时，便会触发该接口中的回调方法
回调方法	public boolean onKey(View v, int keyCode, KeyEvent event) 说明：参数 v 为事件的事件源控件； 参数 keyCode 为手机键盘的键盘码； 参数 event 便为键盘事件封装类的对象，其中包含了事件的详细信息，如发生的事件、事件的类型等

5）OnTouchListener 接口

功能	OnTouchListener 接口用来处理手机屏幕事件的监听接口，当为 View 的范围内触摸按下、抬起或滑动等动作时都会触发该事件
回调方法	public boolean onTouch(View v, MotionEvent event) 说明：参数 v 同样为事件源对象； 参数 event 为事件封装类的对象，其中封装了触发事件的详细信息，同样包括事件的类型、触发时间等信息



6）OnCreateContextMenuListener 接口

功能	OnCreateContextMenuListener 接口用来处理上下文菜单显示事件的监听接口，该方法是定义和注册上下文菜单的另一种方式。
回调方法	public void onCreateContextMenu(ContextMenu menu, View v, ContextMenuInfo info) 说明：参数 menu 为事件的上下文菜单； 参数 v 为事件源 View，当该 View 获得焦点时才可能接收该方法的事件响应； 参数 info：info 对象中封装了有关上下文菜单额外的信息，这些信息取决于事件源 View



6.2.2 基于回调机制的事件处理

回调机制实质上就是将事件的处理绑定在控件上，由图形用户界面控件自己处理事件，回调机制需要自定义 View 来实现。Android 平台中，每个 View 都有自己的处理事件的回调方法，开发人员可以通过重写 View 中的这些回调方法来实现需要的响应事件。当某个事件没有被任何一个 View 处理时，便会调用 Activity 中相应的回调方法。

回调方法如下。

1）onKeyDown 方法

功能	onKeyDown 方法，该方法是接口 KeyEvent.Callback 中的抽象方法，所有的 View 全部实现了该接口并重写了该方法，该方法用来捕捉手机键盘被按下的事件
方法声明	public boolean onKeyDown (int keyCode, KeyEvent event) 参数 keyCode：该参数为被按下的键值即键盘码，手机键盘中每个按钮都会有其单独的键盘码，在应用程序都是通过键盘码才知道用户按下的是哪个键。 参数 event：该参数为按键事件的对象，其中包含了触发事件的详细信息，例如事件的状态、事件的类型、事件发生的时间等。当用户按下按键时，系统会自动将事件封装成 KeyEvent 对象供应用程序使用。 返回值：该方法的返回值为一个 boolean 类型的变量，当返回 true 时，表示已经完整地处理了这个事件，并不希望其他的回调方法再次进行处理， 而当返回 false 时，表示并没有完全处理完该事件，更希望其他回调方法继续对其进行处理，如 Activity 中的回调方法

2）onKeyUp 方法

功能	onKeyUp 方法同样是接口 KeyEvent.Callback 中的一个抽象方法，并且所有的 View 同样全部实现了该接口并重写了该方法，onKeyUp 方法用来捕捉手机键盘按键抬起的事件
方法声明	public boolean onKeyUp (int keyCode, KeyEvent event) 参数 keyCode：参数 keyCode 同样为触发事件的按键码，注意，同一个按键在不同型号的手机中的按键码可能不同。 参数 event：参数 event 同样为事件封装类的对象，其含义与 onKeyDown 方法中的完全相同，在此不再赘述。 返回值：该方法返回值表示的含义与 onKeyDown 方法相同，同样通知系统是否希望其他回调方法再次对该事件进行处理

3）onTouchEvent 方法

功能	onTouchEvent 方法在 View 类中的定义，并且所有的 View 子类全部重写了该方法，应用程序可以通过该方法处理手机屏幕的触摸事件

方法声明	public boolean onTouchEvent (MotionEvent event) 参数 event：参数 event 为手机屏幕触摸事件封装类的对象，其中封装了该事件的所有信息，如触摸的位置、触摸的类型及触摸的时间等，该对象会在用户触摸手机屏幕时被创建。 返回值：该方法的返回值机理与键盘响应事件的相同，同样是当已经完整地处理了该事件且不希望其他回调方法再次处理时返回 true，否则返回 false。 该方法并不像之前介绍过的方法只处理一种事件，一般情况下以下三种情况的事件全部由 onTouchEvent 方法处理，只是三种情况中的动作值不同。 屏幕被按下：当屏幕被按下时，会自动调用该方法来处理事件，此时 Motion Event.getAction()的值为 MotionEvent.ACTION_DOWN，如果在应用程序中需要处理屏幕被按下的事件，只需重新该回调方法，然后在方法中进行动作的判断即可。 屏幕被抬起：当触控笔离开屏幕时触发的事件，该事件同样需要 onTouchEvent 方法来捕捉，然后在方法中进行动作判断。当 MotionEvent.getAction()的值为 MotionEvent.ACTION_UP 时，表示是屏幕被抬起的事件。 在屏幕中拖动：该方法还负责处理触控笔在屏幕上滑动的事件，同样是调用 MotionEvent.getAction()方法来判断动作值是否为 MotionEvent.ACTION_MOVE 再进行处理

4）onTrackBallEvent 方法

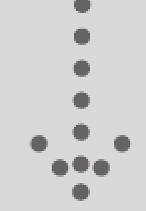
功能	onTrackBallEvent 方法为手机中轨迹球的处理方法，所有的 View 同样全部实现了该方法
方法声明	public boolean onTrackballEvent (MotionEvent event) 参数 event：参数 event 为手机轨迹球事件封装类的对象，其中封装了触发事件的详细信息，同样包括事件的类型、触发时间等，一般情况下，该对象会在用户操控轨迹球时被创建。 返回值：该方法的返回值与前面介绍的各回调方法的返回值机制完全相同

6.2.3 回调方法应用案例

本节通过一个简单的案例介绍 onTouchEvent 方法，其他回调方法的使用方式类似，读者可以自行掌握。本实例实现了在用户单击的位置绘制一个矩形，然后监测用户触控笔的状态，当用户在屏幕上移动触控笔时，使矩形随之移动，而当用户触控笔离开手机屏幕时，停止绘制矩形，其开发步骤如下。

（1）创建一个名为 DrawRactangle 的 Android 项目。打开 DrawRactangle.java 文件，输入如下所示的代码。

```
package wyf.ytl;                    //声明所在包
import android.app.Activity;        //引入 Activity 类
import android.content.Context;     //引入 Context 类
import android.graphics.Canvas;     //引入 Canvas 类
import android.graphics.Color;      //引入 Color 类
import android.graphics.Paint;      //引入 Paint 类
import android.os.Bundle;           //引入 Bundle 类
import android.view.MotionEvent;    //引入 MotionEvent 类
import android.view.View;           //引入 View 类
public class DrawRactangle extends Activity {
```



```
    MyView myView;      //自定义 View 的引用
    public void onCreate(Bundle savedInstanceState)
    {  //重写的 onCreate 方法，该方法会在此 Activity 创建时被系统调用，在方法中先初始化自定义的 View，然后将当前的用户界面设置成该 View
    super.onCreate(savedInstanceState);
    myView = new MyView(this);      //初始化自定义的 View
    setContentView(myView);         //设置当前显示的用户界面
    }
    @Override
    public boolean onTouchEvent(MotionEvent event)
{//onTouchEvent 回调方法重写的屏幕监听方法，在该方法中，根据事件动作的不同执行不同的操作
    switch(event.getAction()){
//当前事件为屏幕被按下的事件，通过调用 MotionEvent 的 getX()和 getY()方法得到事件发生的坐标，然后设置给自定义 View 的 x 与 y 成员变量
    case MotionEvent.ACTION_DOWN:          //按下
      myView.x = (int) event.getX();       //改变 x 坐标
      myView.y = (int) event.getY()-52;    //改变 y 坐标
     myView.postInvalidate();              //重绘
      break;
//表示在屏幕上滑动时的事件，同样是得到事件发生的位置并设置给 View 的 x、y。需要注意的是，因为此时手机屏幕并不是全屏模式，所以需要对坐标进行调整
    case MotionEvent.ACTION_MOVE:          //移动
     myView.x = (int) event.getX();        //改变 x 坐标
      myView.y = (int) event.getY()-52;    //改变 y 坐标
      myView.postInvalidate();             //重绘
      break;
//以下是屏幕被抬起的事件，此时将 View 的 x、y 成员变量设成-100。表示并不需要在屏幕中绘制矩形
    case MotionEvent.ACTION_UP:            //抬起
      myView.x = -100;                     //改变 x 坐标
      myView.y = -100;                     //改变 y 坐标
      myView.postInvalidate();             //重绘
      break;
    }
    return super.onTouchEvent(event);
  }
  class MyView extends View{               //自定义的 View
      Paint paint;                         //画笔
      int x = 50;                          //x 坐标
      int y = 50;                          //y 坐标
      int w = 80;                          //矩形的宽度
    public MyView(Context context)
     { //构造器
     super(context);
      paint = new Paint();                 //初始化画笔
    }
   @Override
    protected void onDraw(Canvas canvas)
    {  //绘制方法
      canvas.drawColor(Color.GRAY);        //绘制背景色
      canvas.drawRect(x, y, x+w, y+w, paint);  //根据成员变量绘制矩形
```

```
            super.onDraw(canvas);
        }
    }
}
```

（2）自定义的 View 并不会自动刷新，所以每次改变数据模型时都需要调用 postInvalidate 方法进行屏幕的刷新操作。运行该案例，将看到如图 6-1 所示的效果。

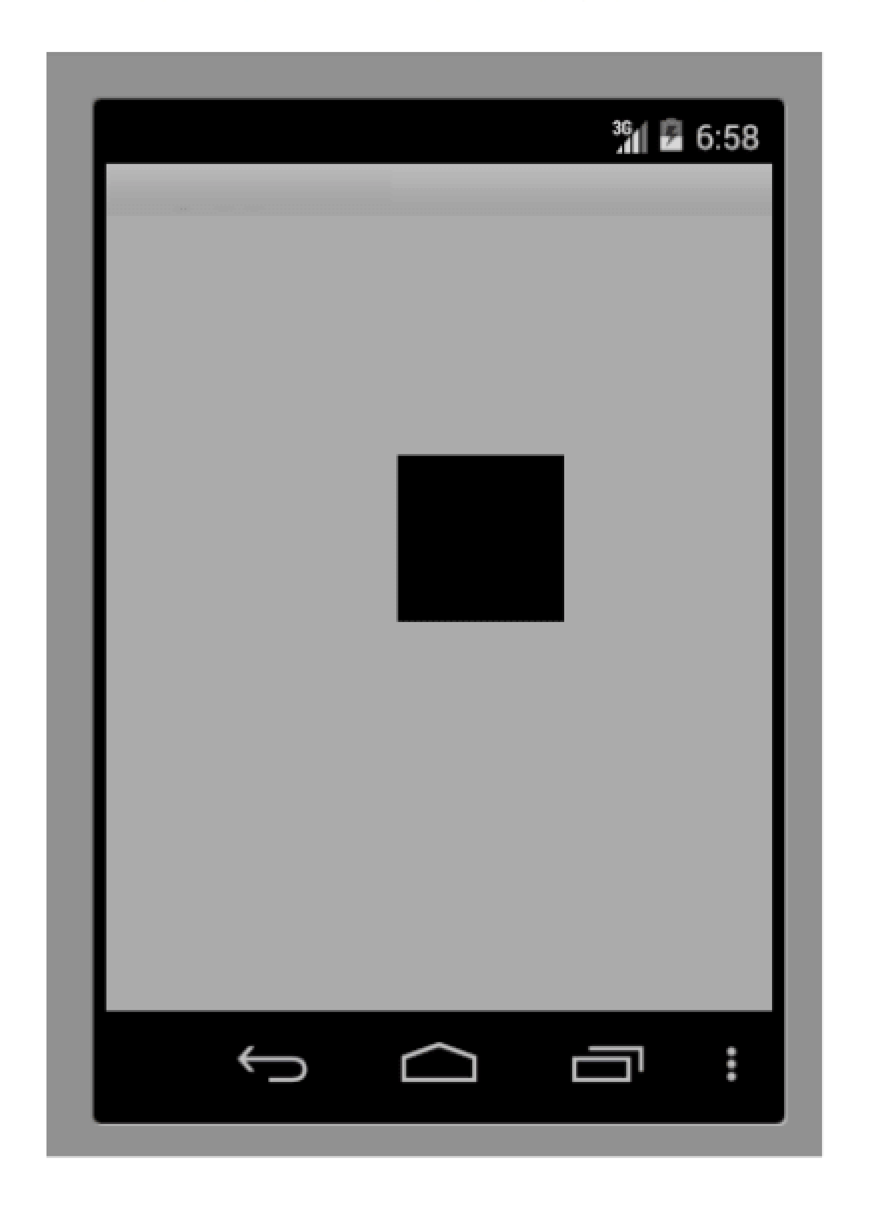
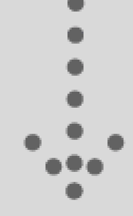


图 6-1　案例效果

单击屏幕时，会在单击的位置绘制一个矩形，当触控笔在屏幕中滑动时，该矩形会随之移动，而当触控笔离开屏幕时，便会取消绘制矩形。

6.3　本章小结

本章主要介绍了多线程机制和事件处理机制，多线程机制是系统开发有效配置资源和提高执行效率的有效手段，而事件处理机制则是实现用户与系统有效交互的必要手段。



第 7 章　2D 应用程序开发

良好的视觉效果一直是衡量用户体验的一个重要指标，其在游戏开发和动画设计中尤为重要，如何实现从静态到动态的图形显示，如何设计简单的可视化游戏，如何实现动画效果，将是本章介绍的内容。

7.1　SurfaceView

SurfaceView 是非常重要的绘图容器。它可以直接从内存或者 DMA 等硬件接口取得图像数据，在主线程之外的线程中向屏幕绘图，在新线程中更新画面，避免画图任务繁重的时候造成主线程 UI 阻塞。

7.1.1　SurfaceView 简介

SurfaceView 是 View 的继承类，View 中内嵌了一个专门用于绘制的 Surface，其格式、尺寸和绘制位置都是可以被控制的。SurfaceView 和 View 最本质的区别在于：使用 SurfaceView 可以在一个新起的单独线程中重新绘制画面，而 View 则必须在 UI 的主线程中更新画面。

以游戏应用为例，View 和 SurfaceView 分别应用于以下情况。

- **被动更新画面**　如果游戏需要用户操作来触发页面更新，即需要简单交互，如棋类游戏等，此时 View 适用。
- **主动更新画面**　如果游戏中有持续的动画行为，如背景的持续更新，或者动画人物一直在运动等。这需要一个单独的线程不停地绘制人物的状态，为了避免阻塞 UI 主线程，需要使用 SurfaceView 来控制。

Surface 是纵深排序（Z-ordered）的，即它总在自己所在窗口的后面。SurfaceView 提供了一个可见区域，只有在可见区域内的 Surface 内容才可见。Surface 的排版显示受视图层级关系的影响，它的兄弟视图结点会在顶端显示。这意味着 Surface 的内容会被它的兄弟视图遮挡，这一特性可以用来放置遮盖物（如文本和按钮等控件）。但是，如果 Surface 上有透明控件，那么它的每次变化都会引起框架重新计算它和顶层控件的透明效果。

每个 Surface 都会创建一个 Canvas 对象，用来管理 View 在 Surface 上的绘图操作，它支持所有标准 Canvas 方法绘图，同时也支持 OpenGLES 库。Surface 会在 SurfaceView 显示之后被创建，在 SurfaceView 隐藏之前被销毁。SurfaceView 可以直接访问一个画布（Canvas），它是提供给需要直接画像素而不是使用窗体部件的应用使用的。

7.1.2 SurfaceView 的使用

访问 Surface 需要通过 SurfaceHolder 接口，使用 SurfaceView.getHolder 方法可以得到 SurfaceHolder 接口对象。SurfaceHolder 接口的几个常用方法如下。

- **addCallback(SurfaceHolder.Callback callback)** 为当前的 SurfaceView 设置一个回调对象，SurfaceView 将会在 Surface 改变时调用该回调。
- **getSurface()** 获得一个可直接访问的 Surface 对象。
- **lockCanvas()** 锁定 Surface 并返回一个 Canvas 对象，其大小为整个 View。
- **unlockCanvasAndPost(Canvas canvas)** 释放指定 Surface 对象上的锁，并把 Canvas 对象发送到 SurfaceView 进行画面更新。
- **lockCanvas(Rect dirty)** 锁定 Surface 并返回一个 Canvas 对象，其大小为指定矩形，只对失效区域进行重绘，可以提高速度。

为了保证可以正确地操作 Canvas 对象，对 Canvas 的任何操作都必须要在 Surface 被创建之后和销毁之前执行，否则将不能获取正确的 Surface 和 Canvas 对象，返回值为 null。此时，需要使用 SurfaceHolder.Callback 接口，Surface 会在自己状态发生变化时通知该接口。该接口中有三个抽象方法，这三个抽象方法是 SurfaceView 的三个生命周期。

- **surfaceCreated(SurfaceHolder holder)** surface 创建时调用，一般在该方法中启动绘图的线程。
- **surfaceChanged(SurfaceHolder holder, int format, int width,int height)** surface 尺寸发生改变时调用，如横竖屏切换。
- **surfaceDestroyed(SurfaceHolder holder)** surface 被销毁时调用，如退出游戏画面时，一般在该方法中停止绘图线程。

7.2 用 2d 技术开发简单游戏

本小节使用一个小的球类游戏来介绍 2d 技术开发，本实例主要用到 SurfaceView、贴图技术及声音处理技术。涉及的处理方法主要有以下 5 个类。

- **BallGameActivity** 设置屏幕的相关属性。
- **GameSurfaceView** 显示界面的设置。
- **ThreadForDraw** 刷帧线程重新绘制游戏界面。
- **ThreadForGo** 控制小球的移动。
- **ThreadForTimecControl** 计算小球运行的时间。

下面分别介绍每个类的实现方法。

1）BallGameActivity[Java]

```
package com.chapter8.pb;
import java.util.HashMap;
import org.apache.http.auth.AUTH;
```

```
import android.app.Activity;
import android.content.Context;
import android.media.AudioManager;
import android.media.SoundPool;
import android.os.Bundle;
import android.view.Window;
import android.view.WindowManager;

public class BallGameActivity extends Activity {
     GameSurfaceView gameSurfaceView;
     SoundPool soundPool;  //建立声音缓冲池
     HashMap<Integer, Integer> soundpoolMap;   //存放声音 ID 的 map
     @Override
     public void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          initSounds();  //初始化声音
          requestWindowFeature(Window.FEATURE_NO_TITLE);
          getWindow().setFlags(WindowManager.LayoutParams.FLAG
          FULLSCREEN,
                   WindowManager.LayoutParams.FLAG FULLSCREEN);//设置全屏
          gameSurfaceView = new GameSurfaceView(this);
          setContentView(gameSurfaceView);
          playSound(1, -1);   //循环播放音乐
     }
     public void playSound(int sound, int loop) {
          AudioManager audioManager = (AudioManager) this
                  .getSystemService(Context.AUDIO_SERVICE);//播放声音的方法
          float streamVolumeMax = audioManager
                  .getStreamMaxVolume(audioManager.STREAM MUSIC);
          float streamVolumeCurrent = audioManager
                  .getStreamVolume(AudioManager.STREAM MUSIC);
          float volume = streamVolumeCurrent / streamVolumeMax;//控制音量
          soundPool.play(soundpoolMap.get(sound), volume, volume, 1, loop,
         0.5f);  //参数声音资源的 ID 左声道右声道优先级循环次数回访速度
     }
     public void initSounds() {
          soundPool = new SoundPool(4, AudioManager.STREAM_MUSIC, 100);
         //参数播放的个数音频类型播放的质量
         soundpoolMap = new HashMap<Integer, Integer>();//创建声音资源的 MAP
         soundpoolMap.put(1, soundPool.load(this, R.raw.bg, 1));
        //将加载声音的资源放在 Map 中
     }
}
```

第一个类主要实现了游戏屏幕的相关属性，主要是声音的配置，为游戏提供好的音频效果。

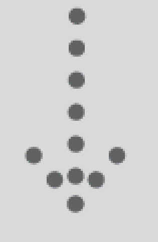
2）GameSurfaceView[html]

```
package com.chapter8.pb;
import android.graphics.Bitmap;
```

```
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.view.MotionEvent;
import android.view.SurfaceHolder;
import android.view.SurfaceView;

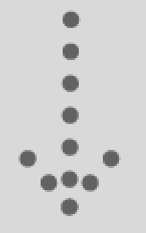
public class GameSurfaceView extends SurfaceView implements
          SurfaceHolder.Callback {
     BallGameActivity ballGameActivity;
     ThreadForTimecControl threadForTimecControl;
     ThreadForGo threadForGo;
     ThreadForDraw threadForDraw;
     int backSize = 14;              //设置背景块大小
     int ScreenWidth = 350;          //设置屏幕宽度
     int screenHeight = 500;         //设置屏幕高度
     int bannerWidth = 40;           //设置挡板宽度
     int bannerHeight = 6;           //设置挡板高度
     int bottomSpance = 15;          //设置下端留白
     int bannerSpan = 5;             //设置板每次移动的距离
     int ballSpan = 9;               //设置球每次移动的距离
     int ballSize = 15;              //设置小球大小
     int hintWidth = 80;             //设置游戏说明宽度
     int hintHeight = 20;            //设置游戏说明高度
     int status = 0;
     //设置游戏状态控制 0：等待开始 1：正在进行 2：游戏结束 3：游戏胜利
     int score = 0;                  //设置得分初始值为 0
     int ballx;                      //设置小球 x 坐标
     int bally;                      //设置小球 y 坐标
     int direction = 0;              //设置小球方向
     int bannerX;                    //设置挡板 X 坐标
     int bannerY;                    //设置挡板 Y 坐标
     int scoreWidth = 32;
     Bitmap iback;                   //背景图
     Bitmap[] iscore = new Bitmap[10];//得分图
     Bitmap iball;                   //小球位图
     Bitmap ibanner;                 //挡板位图
     Bitmap ibegin;                  //开始位图
     Bitmap igameover;               //游戏结束位图
     Bitmap iwin;                    //游戏结束位图
     Bitmap iexit;                   //退出位图
     Bitmap ireplay;                 //重玩位图

     public GameSurfaceView(BallGameActivity ballGameActivity) {
          super(ballGameActivity);
          getHolder().addCallback(this);  //注册回调接口
          this.ballGameActivity = ballGameActivity;
          initBitmap();
          threadForDraw = new ThreadForDraw(this);
     }
     public void initBitmap() {
          iback = BitmapFactory.decodeResource(getResources(), R.drawable.
```



```
            back);
        iscore[0] = BitmapFactory.decodeResource(getResources(), R.
        drawable.d0);
        iscore[1] = BitmapFactory.decodeResource(getResources(), R.
        drawable.d1);
        iscore[2] = BitmapFactory.decodeResource(getResources(), R.
        drawable.d2);
        iscore[3] = BitmapFactory.decodeResource(getResources(), R.
        drawable.d3);
        iscore[4] = BitmapFactory.decodeResource(getResources(), R.
        drawable.d4);
        iscore[5] = BitmapFactory.decodeResource(getResources(), R.
        drawable.d5);
        iscore[6] = BitmapFactory.decodeResource(getResources(), R.
        drawable.d6);
        iscore[7] = BitmapFactory.decodeResource(getResources(), R.
        drawable.d7);
        iscore[8] = BitmapFactory.decodeResource(getResources(), R.
        drawable.d8);
        iscore[9] = BitmapFactory.decodeResource(getResources(), R.
        drawable.d9);
        iball = BitmapFactory.decodeResource(getResources(),R.drawable.
        ball);
        ibanner = BitmapFactory.decodeResource(getResources(),
                  R.drawable.banner);
        ibegin = BitmapFactory.decodeResource(getResources(), R.
        drawable.begin);
        igameover = BitmapFactory.decodeResource(getResources(),
                  R.drawable.gameover);
        iwin = BitmapFactory.decodeResource(getResources(), R. drawable.
        win);
        iexit = BitmapFactory.decodeResource(getResources(), R. drawable.
        exit);
        ireplay = BitmapFactory.decodeResource(getResources(),
                  R.drawable.replay);
        initBallAndBanner();  //初始化小球位置及板 X 坐标
    }
    public void initBallAndBanner() {
        bally = screenHeight - bottomSpance - bannerHeight - ballSize;
        ballx = screenWidth / 2 - ballSize / 2;  //初始化小球位置
        bannerX = screenWidth / 2 - bannerWidth / 2;
        bannerY = screenHeight - bottomSpance - bannerHeight;
       //初始化板 X，Y 坐标
    }
    public void replay() {
        if (status == 2 || status == 3) {
            initBallAndBanner();  //初始化小球的位置
            score = 0;
            status = 0;
            direction = 3;
        }
```

```
}
@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    //清除背景
    int colum = screenWidth / backSize
            + ((scoreWidth % backSize == 0) ?0 : 1);
    int rows = screenHeight / backSize
            + ((screenHeight % backSize == 0) ? 0 : 1);
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < colum; j++) {
            canvas.drawBitmap(iback, 16*j, 16*i, null);
        }
    }
    String scorestr = score + "";  //绘制得分
    int loop = 3 - scorestr.length();
    for (int i = 0; i < loop; i++) {
        scorestr = "0" + scorestr;
    }
    int startX = screenWidth - scoreWidth * 3 - 10;
    for (int i = 0; i < 3; i++) {
        int tempScore = scorestr.charAt(i) - '0';
      canvas.drawBitmap(iscore[tempScore], startX + i * scoreWidth,
      5, null);
    }
    canvas.drawBitmap(iball, ballx, bally, null);  //绘制小球
    canvas.drawBitmap(ibanner, bannerX, bannerY, null);  //绘制板
    if (status == 0) {
        canvas.drawBitmap(ibegin, screenWidth / 2 - hintWidth / 2,
                screenHeight / 2 - hintHeight / 2, null);
    }  //绘制开始提示
    if (status == 2) {
        canvas.drawBitmap(igameover, screenWidth/2 - hintWidth /2,
          screenHeight / 2 - hintHeight / 2, null);
    }  //绘制失败提示
    if (status == 3) {
        canvas.drawBitmap(iwin, screenWidth / 2 - hintWidth / 2,
                screenHeight / 2 - hintHeight / 2, null);
    }  //绘制胜利提示
    canvas.drawBitmap(iexit, screenWidth - 32, screenHeight - 16,
    null);  //绘制退出提示
    if (status == 2 || status == 3) {
        canvas.drawBitmap(ireplay, 0, screenHeight - 16, null);
    }   ///绘制重玩提示
}
@Override
public boolean onTouchEvent(MotionEvent event) {
     int x = (int) event.getX();
     int y = (int) event.getY();
     if (x < screenWidth && x > screenWidth - 32 && y < screenHeight
     && y > screenHeight - 16) {
```



```
                    ballGameActivity.soundPool.stop(1);
                    System.exit(0);  //按下退出选项退出系统
              }
              if (status == 0) {  //等待状态
                    status = 1;
                    threadForTimecControl = new ThreadForTimecControl(this);
                    threadForGo = new ThreadForGo(this);
                    threadForTimecControl.start();
                    threadForGo.start();
              } else if (status == 1) {
                    bannerX = x;
              } else if (status == 2 || status == 3) {
                    if (x < 32 && x > 0 && y < screenHeight && y > screenHeight
                    - 16) {
                         replay();    //按下重玩
                    }
              }
              return super.onTouchEvent(event);
        }
        @Override
        public void surfaceChanged(SurfaceHolder holder, int format, int width,
                  int height) {
        }
        @Override
        public void surfaceCreated(SurfaceHolder holder) {
             this.threadForDraw.flag = true;
             threadForDraw.start();    //创建时候启动相关进程
        }
        @Override
        public void surfaceDestroyed(SurfaceHolder holder) {
             boolean retry = true;  //释放相应的进程
             this.threadForDraw.flag = false;
             while (retry) {
                  try {
                        threadForDraw.join();
                        retry = false;
                  } //不断循环直到刷帧结束
             catch (InterruptedException e) {
                 }
             }
        }
    }
```

第二个类为显示界面的设置，分别设定并初始化背景块、屏幕、挡板和小球的基本属性，建造游戏进行的背景框架和必要元素。

3）ThreadForDraw[Java]

```
    package com.chapter8.pb;
    import android.graphics.Canvas;
    import android.view.SurfaceHolder;
    public class ThreadForDraw extends Thread {
```

```
boolean flag=true;
int sleep=100;
GameSurfaceView gameSurfaceView;
SurfaceHolder surfaceHolder;
public ThreadForDraw (GameSurfaceView gameSurfaceView){
     this.gameSurfaceView=gameSurfaceView;
     this.surfaceHolder=gameSurfaceView.getHolder();
}
@Override
public void run() {
    //TODO Auto-generated method stub
    Canvas canvas;
    while (this.flag) {
         canvas=null;
         try {
              canvas=this.surfaceHolder.lockCanvas(null);
              synchronized (this.surfaceHolder) {
                  gameSurfaceView.onDraw(canvas);
             }
         } catch (Exception e) {
         }finally{
              if (canvas!=null) {
                    this.surfaceHolder.unlockCanvasAndPost(canvas);
              }
         }
         try {
              Thread.sleep(sleep);
             }
         catch (Exception e) {
           }
    }
    super.run();
}
}
```

第三个类主要用到本节的 surfaceview 刷新帧线程重新绘制游戏界面。

4）ThreadForGo[Java]

```
package com.chapter8.pb;
//游戏过程中移动小球的线程
public class ThreadForGo extends Thread {
     boolean flag = true;               //设置线程执行的标志
     GameSurfaceView gameSurfaceView;  //游戏界面的引用
     public ThreadForGo(GameSurfaceView gameSurfaceView) {
          this.gameSurfaceView = gameSurfaceView;
     }
     @Override
     public void run() {
          //TODO Auto-generated method stub
          while (flag) {
               switch (gameSurfaceView.direction) {
```




```
        case 0:                    //右上控制当前方向移动球
            gameSurfaceView.ballx = gameSurfaceView.ballx
                    + gameSurfaceView.ballSpan;
            gameSurfaceView.bally = gameSurfaceView.bally
                    - gameSurfaceView.ballSpan;
            //判断是否碰壁
            if (gameSurfaceView.ballx >= gameSurfaceView.
            screenWidth
                    - gameSurfaceView.ballSize) {
                gameSurfaceView.direction = 3;
            }  //碰到上壁
     else if (gameSurfaceView.bally <= 0) {
                gameSurfaceView.direction = 1;
            }
            break;
        case 1:                    //右下
            gameSurfaceView.ballx = gameSurfaceView.ballx
                    + gameSurfaceView.ballSpan;
            gameSurfaceView.bally = gameSurfaceView.bally
                    + gameSurfaceView.ballSpan;

            if (gameSurfaceView.bally >= gameSurfaceView.
            screenHeight
                     - gameSurfaceView.bannerHeight
                     - gameSurfaceView.bottomSpance
                     - gameSurfaceView.ballSize) {
                checkCollision(1);
            }  //碰到下壁
      else if (gameSurfaceView.ballx >= gameSurfaceView. screenWidth
                     - gameSurfaceView.ballSize) {
                gameSurfaceView.direction = 2;
            }    //碰到右壁
            break;
        case 2:
            //左下
            gameSurfaceView.ballx = gameSurfaceView.ballx
                     - gameSurfaceView.ballSpan;
            gameSurfaceView.bally = gameSurfaceView.bally
                     + gameSurfaceView.ballSpan;
            if (gameSurfaceView.bally >= gameSurfaceView.
            screenHeight
                     - gameSurfaceView.bannerHeight
                     - gameSurfaceView.bottomSpance
                     - gameSurfaceView.ballSize) {
                checkCollision(2);
            }               //碰到下壁
            else if (gameSurfaceView.ballx <= 0) {
                gameSurfaceView.direction = 1;
            }    //碰到左壁
            break;
        case 3:
```

```
                        gameSurfaceView.ballx = gameSurfaceView.ballx
                                - gameSurfaceView.ballSpan;
                        gameSurfaceView.bally = gameSurfaceView.bally
                                - gameSurfaceView.ballSpan;
                        if (gameSurfaceView.ballx <= 0) {
                              gameSurfaceView.direction = 0;
                        } ///碰到左壁
                        else if (gameSurfaceView.bally <= 0) {
                               gameSurfaceView.direction = 2;
                        }   //碰到上壁
                        break;
                    default:
                        break;
                    }
                    try {
                        Thread.sleep(100);
                    } catch (Exception e) {
                         //TODO: handle exception
                    }
              }
        }
        public void checkCollision(int dirction) {
             if (gameSurfaceView.ballx >= gameSurfaceView.bannerX
                       - gameSurfaceView.ballSize
                       && gameSurfaceView.ballx <= gameSurfaceView.bannerX
                                 + gameSurfaceView.bannerWidth) {
                 switch (dirction) {
                 case 1:
                      gameSurfaceView.direction = 0;
                      break;
                 case 2:
                      gameSurfaceView.direction = 3;
                      break;
                 default:
                      break;
                 }
          } else {
                 //没有碰到板
                 gameSurfaceView.threadForTimecControl.flag = false;
                 gameSurfaceView.threadForGo.flag = false;
                 gameSurfaceView.status = 2;
          }
      }
}
```

第四个类控制小球的移动，并通过其位置识别小球的状态，是否碰壁，是否碰到了挡板等。

5）ThreadForTimecControl[Java]

```
package com.chapter8.pb;
```

```
public class ThreadForTimecControl extends Thread {
    //计算生存时间的线程
    //判断是否胜利的值
    int highest = 200;
// 游戏界面的引入
    GameSurfaceView gameSurfaceView;
    //线程标志位
    boolean flag = true;
    public ThreadForTimecControl(GameSurfaceView gameSurfaceView) {
         this.gameSurfaceView = gameSurfaceView;
    }
    @Override
    public void run() {
        //TODO Auto-generated method stub
        while(flag){
                gameSurfaceView.score++;
                if (gameSurfaceView.score==highest) {
                      //游戏胜利
                      gameSurfaceView.status=3;
                      gameSurfaceView.threadForTimecControl.flag=false;
                      gameSurfaceView.threadForGo.flag=false;
                }
                try {
                      Thread.sleep(1000);
                } catch (Exception e) {
                     //TODO: handle exception
                }
          }
      }
}
```

第五个类通过计算小球的生存时间判断游戏是否获胜。

通过这五个类的配合可以实现小球的控制和判断是否得分，并且设定了比较完整的界面，完成了一个小型球类的操作。

7.3 Graphics 类开发

Android 中需要通过 Graphics 类来显示 2D 图形。Graphics 中包含了 Canvas（画布）、Paint（画笔）、Color（颜色）、Bitmap（图像）、2D 几何图形等常用类。Graphics 具有绘制点、线、颜色、图像处理、2D 几何图形等功能。

Paint 类是 Android 中的画笔，有很多方法设置其属性，主要的方法如下。

- **setAntiAlias**　设置画笔的锯齿效果。
- **setColor**　设置画笔的颜色。
- **setARGB**　设置画笔的 argb 值。
- **setAlpha**　设置透明度。

- **setTextSize** 设置字体尺寸。
- **setStyle** 设置画笔的风格，空心或实心。
- **setStrokeWidth** 设置空心的边框宽度。
- **getColor** 得到画笔的颜色。
- **getAlpha** 得到画笔的透明度。

Color 类主要定义了一些颜色常量，以及对颜色的转换等。

Canvas 类是画布，可以在画布上绘制想要的东西，其提供了一些方法。

- **Canvas()** 创建一个空的画布，可以使用 setBitmap 方法来设置绘制的具体画布。
- **Canvas(Bitmap bitmap)** 以 bitmap 对象创建一个画布，则将内容都绘制在 bitmap 上，隐藏 bitmap 不得为 NULL。
- **Canvas(GL gl)** 在绘制 3D 效果时使用，与 OpenGL 相关。
- **drawColor** 设置 Canvas 的背景颜色。
- **setBitmap** 设置具体画布。
- **clipRect** 设置显示区域，即设置裁剪区。
- **isOpaque** 检测是否支持透明。
- **rotate** 旋转画布。旋转画布时会旋转画布上所有对象，若只要旋转一个，需要用到 save 方法锁定需要操作的对象，在操作之后通过 restore 方法来解锁。
- **setViewPort** 设置画布中显示的窗口。
- **skew** 设置偏移量。

Android 中可以绘制的几何图形如下。

- **drawRect** 绘制矩形。
- **drawCircle** 绘制圆形。
- **drawOval** 绘制椭圆。
- **drawPath** 绘制任意多边形。
- **drawLine** 绘制直线。
- **drawPoint** 绘制点。

Android 中还可以通过 ShapeDrawable 来绘制图像，ShapeDrawable 可以设置画笔的形状，通过 getPaint 方法可以得到 Paint 对象。在 ShapeDrawable 中提供了 setBounds 方法来设置图形显示的区域，最后通过 ShapeDrawable 的 Draw 方法将图形显示到屏幕上。

Android 中提供一系列的 drawText 方法来绘制字符串，在绘制字符串之前需要设置画笔对象，包括字符串的尺寸、颜色等属性。使用 FontMetrics 来规划字体的属性，可以通过 getFontMetrics 方法来获得系统字体的相关内容。

字符串处理的常用方法如下。

- **setTextSize** 设置字符串的尺寸。
- **setARGB** 设置颜色。
- **getTextWidths** 取得字符串的宽度。
- **setFlags（Paint.ANTI_ALIAS_FLAG)** 消除锯齿。

图像绘制：在 Android 中，“res/drawable”目录用来存放图像资源，Android 中提供了 Bitmap 来存放这些资源。((BitmapDrawable) getResource().getDrawable(ID)).getBitmap()可以

获得图像对象 Bitmap，然后使用 drawBitmap 方法将图像显示到屏幕上（canvas.drawBitmap（bitmap，x，y，null）），Bitmap 还提供了一些方法，如 getHeight 方法获得图像的高度和 getWidth 方法获得图像的宽度。

图像旋转：图像旋转需要使用 Matrix，包含一个 3×3 的矩阵，用于图像变换。Matrix 没有结构体，通过 reset 方法或 set 方法来初始化。通过 setRotate 可以设置旋转角度，用 creatBitmap 可以创建一个经过旋转等处理的 Bitmap 对象，然后将 Bitmap 对象绘制到屏幕上，实现图像旋转操作。

图像缩放：Matrix 的 postScale 方法可以设置图像缩放的倍数。

图像像素操作：Android 中 Bitmap 提供了操作像素的方法，通过 getPixels 方法获得图像的像素并放到一个数组中，操作像素通过处理数组实现，使用 setPixels 设置这个像素数组到 Bitmap 中。

图像渲染：Android 中提供了 Shader 类专门用来渲染图像和一些几何图形，Shader 包含几个直接子类。

- **BitmapShader** 将图片裁剪成椭圆或圆形等形状。
- **ComposeShader** 混合渲染，可以和其他渲染类混合使用。
- **LinearGradient** 线性渐变。
- **RadialGradient** 环形渐变。
- **SweepGradient** 梯度渐变。

Shader 类的使用，都需要先构建 Shader 对象，然后通过 Paint 的 setShader 方法来设置渲染对象，在绘制时使用 Paint 对象即可。

双缓冲技术：如果程序在动画正在显示时需要重新绘制图像，由于之前的动画画面还未显示完成，屏幕会出现不停闪烁的现象。而使用双缓冲技术可以帮助避免闪烁。只需将要处理的图片在内存中处理好，再将其显示到屏幕上，这样显示出来的总是完整的图像，而不会出现闪烁现象。Android 中的 SurfaceView 就使用了双缓冲机制。双缓冲的核心技术是先通过 setBitmap 方法将要绘制的所有图形绘制到一个 Bitmap 上，然后调用 drawBitmap 方法绘制出这个 Bitmap，显示在屏幕上。

全屏显示：通过 requestWindowFeature 方法设置标题栏是否显示，通过 setFlags 方法设置全屏模式。

获得屏幕属性：Android 中的 DisplayMetrics 定义了屏幕的一些属性，可通过 getMetrics 方法得到当前屏幕的 DisplayMetrics 属性，获得屏幕的宽和高。

7.4 动画实现

Android 提供了三种动画效果，分别是逐帧动画（frame-by-frame animation），类似于 GIF 格式的图片，一帧一帧的显示来呈现动画效果；布局动画（layout animation），用来设置 layout 内的所有 UI 控件；控件动画（view animation），可以应用到某个 view 上的动画。

7.4.1 逐帧动画

逐帧动画即通过播放预先排序好的图片来实现动态的画面，像放电影一样。实现步骤如下。

（1）在工程里面导入要播放的图片。例如 icon1，icon2，icon3，如图 7-1 所示。



图 7-1 逐帧动画实现素材 icon1，icon2，icon3

（2）在工程 res 文件目录下新建一个 anima 文件夹，并在文件夹中新建一个 start_animation.xml 格式文件，此文件用来定义动画播放图片的顺序及每一张图片显示和停留时间。

```
<?xml version="1.0" encoding="utf-8"?>
<animation-list xmlns:android="http://schemas.android.com/apk/res/
android"
    android:oneshot="false">
    <item android:drawable="@drawable/ icon1" android:duration="1000" />
    <item android:drawable="@drawable/ icon2" android:duration="500" />
    <item android:drawable="@drawable/ icon3" android:duration="600" />
</animation-list>
```

icon1，icon2，icon3 为依次显示的图片，存放在 drawable-mdpi 文件下，一般 1 秒钟播放 24 张图片（帧）就感觉很流畅了，即 duration 为 40 毫秒左右。

（3）布局文件中添加一个 ImageView 控件，用来播放动画图片，具体布局如下。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout width="match parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    <Button
        android:id="@+id/button1"
        android:layout width="wrap content"
        android:layout height="wrap content"
        android:layout gravity="center"
        android:text="开始" />
    <Button
        android:id="@+id/button2"
```

```
        android:layout width="wrap content"
        android:layout height="wrap content"
          android:layout gravity="center"
        android:text="结束" />
    <ImageView
        android:id="@+id/image"
        android:background="@anim/start animation"
        android:layout width="fill parent"
        android:layout height="fill parent"/>
</LinearLayout>
```

（4）以下为实现代码，具体实现效果如图 7-2 所示。



图 7-2　逐帧动画实现

```
public class TestActivity extends Activity{
AnimationDrawable anim;
public void onCreate(Bundle savedInstanceState){
super.onCreate(savedInstanceState);
setContentView(R.layout.start screen);
ImageView image = (ImageView) findViewById(R.id.image);
//image.setBackgroundResource(R.anim.start_animation);指定播放的资源图片
anim = (AnimationDrawable) image.getBackground();
Button start = (Button) findViewById(R.id.button1);
Button stop = (Button) findViewById(R.id.button2);
start.setOnClickListener(new OnClickListener(){
public void onClick(View arg0){
anim.start();
}
};
stop.setOnClickListener(new OnClickListener(){
public void onClick(View arg0){
anim.stop();
}
};
```

```
}}
```

逐帧动画适合复杂的、每一帧图像都有较大变化的动画，但其缺点是需要空间存放大量的图片，所以一般这种方式并不常用。

7.4.2 布局动画

布局动画是渐变动画，Android 通过改变 UI 的属性（大小、位置、透明度等）来实现动画效果，每个 view 都对应一个矩阵来控制该 view 显示的位置，通过不同的方式来改变该控制矩阵就可以实现动画效果，如旋转、移动、缩放等。

android.view.animation.Animation 类代表所有动画变换的基类，其中 5 个实现类如下。

- **AlphaAnimation** 实现 alpha 渐变，可以使界面逐渐消失或者逐渐显现。
- **TranslateAnimation** 实现位置移动渐变，需要指定移动的开始和结束坐标。
- **ScaleAnimation** 实现缩放渐变，可以指定缩放的参考点。
- **RotateAnimation** 实现旋转渐变，可以指定旋转的参考点，默认值为(0,0)左上角。
- **AnimationSet** 代表上面的渐变组合。

还有一个和渐变动画效果关系比较密切的类 android.view.animation.Interpolator，该类定义了渐变动画改变的速率，可以设置为加速变化、减速变化或者重复变化。

下面分别介绍前四个类的实现。

（1）AlphaAnimation：在 res/anima/下新建 alpha_animation.xml 文件，文件如下。

```
<alpha
xmlns:android="http://schemas.android.com/apk/res/android"
android:interpolator="@android:anim/accelerate interpolator"
android:fromAlpha="0.0"
android:toAlpha="1.0"        //指从完全透明 fromAlpha="0.0"到 toAlpha="1.0"完
                              全不透明
android:duration="1000" />
```

（2）TranslateAnimation：在 res/anima/下新建 translate_animation.xml 文件，文件如下。

```
<translate
xmlns:android="http://schemas.android.com/apk/res/android"
android:interpolator="@android:anima/accelerate interpolator"
android:fromYDelta="-100%"
android:toYDelta="0"      //指从 fromYDelta="-100%"的位置下落到 toYDelta="0"
android:duration="500" /> //渐变时间
```

（3）ScaleAnimation：在 res/anima/下新建 scale_animation.xml 文件，文件如下。

```
<scale
xmlns:android="http://schemas.android.com/apk/res/android"
android:interpolator="@android:anima/accelerate_interpolator"
android:fromXScale="1"
android:toXScale="1"        //X 方向不变
android:fromYScale="0.1"
android:toYScale="1.0"      //Y 从 0.1 到 1.0
```




```
    android:duration="500"          //渐变时间
    android:pivotX="50%"            //中心点 X
    android:pivotY="50%"            //中心点 Y
    android:startOffset="100" />    //开始动画的时间
```

（4）RotateAnimation：在 res/anima/下新建 rotate_animation.xml 文件，文件如下。

```
<rotate
xmlns:android="http://schemas.android.com/apk/res/android"
android:interpolator="@android:anima/accelerate interpolator"
android:fromDegrees="0.0"
android:toDegrees="360"      //从 0 度变到 360 度，中心坐标为中心，渐变时间为 500
android:pivotX="50%"
android:pivotY="50%"
android:duration="500" />
```

7.4.3 控件动画

控件动画本质上也是布局动画的一种，可以看作是自定义的动画实现，布局动画在 XML 中定义 OPhone 已经实现的几个动画效果（AlphaAnimation、TranslateAnimation、ScaleAnimation、RotateAnimation），而控件动画就是在代码中继承 android.view.animation.Animation 类来实现自定义效果。

它通过重写 Animation 的 applyTransformation (float interpolatedTime, Transformation t) 函数来实现自定义动画效果，另外一般也会实现 initialize (int width, int height, int parentWidth, int parentHeight)函数，初始化一些相关的参数，如设置动画持续时间、设置 Interpolator、设置动画的参考点等。它是一个回调函数告诉 Animation 目标 View 的大小参数。OPhone 在绘制动画的过程中会反复调用 applyTransformation 函数，每次调用参数 interpolatedTime 值都会变化，该参数从 0 渐变为 1，当该参数为 1 时表明动画结束。通过参数 Transformation 来获取变换的矩阵（matrix），通过改变矩阵就可以实现各种复杂的效果。下面来看一个简单的实现。

```
class ViewAnimation extends Animation {
          public ViewAnimation() {
          }

          @Override
          public void initialize(int width, int height, int parentWidth,
         int parentHeight) {
            super.initialize(width, height, parentWidth, parentHeight);
            setDuration(2500);
            setFillAfter(true);
            setInterpolator(new LinearInterpolator());
         }
         @Override
         protected void applyTransformation(float interpolatedTime,
         Transformation t) {
            final Matrix matrix = t.getMatrix();
```

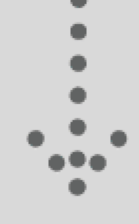
```
            matrix.setScale(interpolatedTime, interpolatedTime);
        }
    }
```

其中，在 Initialize 函数中设置变换持续的时间 2 500 毫秒，然后设置 Interpolator 为 LinearInterpolator，并设置 FillAfter 为 true 这样可以在动画结束的时候保持动画的完整性。在 applyTransformation 函数中通过 MatrixsetScale 函数来缩放，该函数的两个参数代表 X、Y 轴缩放因子，由于 interpolatedTime 是从 0 到 1 变化所在这里实现的效果就是控件从最小逐渐变化到最大。调用 View 的 startAnimation 函数（参数为 Animation）就可以使用自定义的动画了。代码如下（src\org\goodev\animation\ViewAnimActivity.java）。

```
public class ViewAnimActivity extends Activity {
     Button mPlayBtn;
     ImageView mAnimImage;
     @Override
     public void onCreate(Bundle savedInstanceState) {
         super.onCreate(savedInstanceState);
         setContentView(R.layout.view anim layout);
         mAnimImage = (ImageView) this.findViewById(R.id.anim image);
         mPlayBtn = (Button) findViewById(R.id.play btn);
         mPlayBtn.setOnClickListener(new OnClickListener() {
             @Override
             public void onClick(View view) {
                 mAnimImage.startAnimation(new ViewAnimation());
             }
         });
     }
}
```

布局代码如下（res\layout\view_anim_layout.xml）。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout width="fill parent"
android:layout height="fill parent"
>
<Button
android:id="@+id/play btn"
android:layout width="fill parent"
android:layout height="wrap content"
android:text="Start Animation"
/>
<ImageView
android:id="@+id/anim image"
android:persistentDrawingCache="animation|scrolling"
android:layout width="fill parent"
android:layout height="wrap content"
android:src="@drawable/ophone"
```



```
    />
    </LinearLayout>
```

ImageView 是从左上角出来的，这是由于没有指定矩阵的变换参考位置，默认位置为（0,0），如果要想让 ImageView 从中间出来，可以通过矩阵变换来把参考点移动到中间来，实现如下。

```
class ViewAnimation extends Animation {
            int mCenterX;//记录 View 的中间坐标
            int mCenterY;
            public ViewAnimation() {
            }
            @Override
            public void initialize(int width, int height, int parentWidth,
            int parentHeight) {
                 super.initialize(width, height, parentWidth,
                 parentHeight);
                 mCenterX = width/2;
                 mCenterY = height/2;     //初始化中间坐标值
                 setDuration(2500);
                 setFillAfter(true);
                 setInterpolator(new LinearInterpolator());
            }
            @Override
            protected void applyTransformation(float interpolatedTime,
            Transformation t) {
                  final Matrix matrix = t.getMatrix();
                  matrix.setScale(interpolatedTime, interpolatedTime);
                  //通过坐标变换，把参考点（0,0）移动到 View 中间
                  matrix.preTranslate(-mCenterX, -mCenterY);
                  //动画完成后再移回来
                  matrix.postTranslate(mCenterX, mCenterY);
            }
    }
```

preTranslate 函数是在缩放前移动，而 postTranslate 是在缩放完成后移动，现在 ImageView 就是从中间出来的。这样通过操作 Matrix 可以实现各种复杂的变换。操作 Matrix 是实现动画变换的重点，它的常用操作如下。

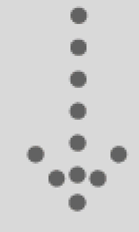
- **Reset()** 重置矩阵。
- **setScale()** 置矩阵缩放。
- **setTranslate()** 设置矩阵移动。
- **setRotate()** 设置矩阵旋转。
- **setSkew()** 矩阵变形（扭曲）。

OPhone 还提供了一个用来监听 Animation 事件的监听接口 AnimationListener，提供三个回调函数：onAnimationStart、onAnimationEnd、onAnimationRepeat，分别对应何时开始、

何时结束、何时重复播放。

7.5　本章小结

本章介绍了二维应用程序开发所用到的关键技术。首先从介绍 SurfaceView 入手，然后通过简单小球游戏的开发实例给出了其应用，在简介 Graphics 类的基础上，详述了动画实现的三种方式。



第 8 章　Android 数据存储

本章主要介绍 Android 平台的数据存储的基础知识，以及学习如何在 Android 应用程序中存储数据。鉴于用户通常需要重用数据，因此在大多数应用软件开发中，存储数据都是一个非常重要的问题。对于 Android 应用程序来说，大体上有三种存储数据的基本方式：使用一种轻量级的机制，即 SharedPreferences 保存少量数据；通过传统的文件系统 API 存储数据到文件中；使用关系型数据库，特别是 SQLite 数据库来存取数据。本章讲解的技术主要用来创建和访问应用程序自身的私有数据。如果想要和其他应用程序进行数据共享的话，需要用到 Content Provider。

8.1　SharedPreferences

本节将主要介绍如何使用 SharedPreferences 来存储数据。Android 平台提供 SharedPreferences 对象来保存简单的应用程序数据。其作用类似于 Windows 平台上常见的 ini 文件，用来保存应用程序的一些配置信息。例如，应用程序可能有一个选项允许用户指定应用中显示文本的字体大小。此时应用程序必须记住用户设置的字体大小，以便下次再次使用该应用程序时，该应用能够合适地把字体设置为用户上次选择的字体大小。其实想要达到这种效果，把配置信息保存到文件和数据库中也是可以的。但是，如果需要保存的配置信息太多，如文本大小、字体、背景色等，那么保存到文件将变得非常麻烦。把配置信息保存在数据库中当然也是可以的，不过把简单数据保存在数据库中有点过分，而且也会影响应用程序的性能。SharedPreferences 对象是通过名值对方式来保存配置信息的，然后把这些名值对写入一个 XML 文件中，相当方便。

案例：使用 SharedPreferences 对象保存和修改配置信息。

通过该案例，读者可以学会如何使用 SharedPreferences 对象存储和修改配置数据，以及如何使用 PreferenceActivity 来显示配置信息。

实现步骤如下。

（1）使用 Eclipse 创建一个新的 Android 项目，命名为 LearningPreferences。

（2）在 res 文件夹中创建一个新的子文件夹，命名为 xml。在新创建的 xml 文件夹中，添加一个新的文件 myappprefs.xml。myappprefs.xml 文件的内容如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen
    xmlns:android="http://schemas.android.com/apk/res/android">
    <PreferenceCategory android:title="Category 1">
        <CheckBoxPreference
            android:title="Checkbox"
            android:defaultValue="false"
```

```
                android:summary="True or False"
                android:key="checkboxPref" />
        </PreferenceCategory>
        <PreferenceCategory android:title="Category 2">
            <EditTextPreference
                android:summary="Enter a string"
                android:defaultValue="[Enter a string here]"
                android:title="Edit Text"
                android:key="editTextPref"
                />
        </PreferenceCategory>
        </PreferenceScreen>
```

（3）创建一个新的类文件，命名为 AppPrefActivity。AppPrefActivity.java 文件的内容如下所示。

```
package com.selfteaching.learningpreferences;

import android.preference.PreferenceActivity;
import android.os.Bundle;

public class AppPrefActivity extends PreferenceActivity {
    @Override
    public void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        //load preferences from the XML file
        addPreferencesFromResource(R.xml.myappprefs);
    }
}
```

（4）在 AndroidManifest.xml 文件中，添加 AppPrefActivity 类的入口，如下面粗体代码所示。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.selfteaching.learningpreferences"
   android:versionCode="1"
   android:versionName="1.0" >

   <uses-sdk android:minSdkVersion="14" />

   <application
      android:icon="@drawable/ic launcher"
      android:label="@string/app name" >
      <activity
         android:label="@string/app name"
         android:name=".LearningPreferencesActivity" >
         <intent-filter >
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
```

```
            </intent-filter>
        </activity>
        <activity android:name=".AppPrefActivity"
                android:label="@string/app_name">
            <intent-filter>
                <action
                    android:name="com.selfteaching.AppPrefActivity" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

（5）修改 main.xml 文件，进行页面布局，内容如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout width="fill parent"
    android:layout height="fill parent"
    android:orientation="vertical" >

<Button
    android:id="@+id/btnPreferences"
    android:text="Load Preferences Screen"
    android:layout width="fill parent"
    android:layout height="wrap content"
    android:onClick="onClickLoad"/>

<Button
    android:id="@+id/btnDisplayValues"
    android:text="Display Preferences Values"
    android:layout width="fill parent"
    android:layout_height="wrap_content"
    android:onClick="onClickDisplay"/>

<EditText
    android:id="@+id/txtString"
    android:layout width="fill parent"
    android:layout height="wrap content" />

<Button
    android:id="@+id/btnModifyValues"
    android:text="Modify Preferences Values"
    android:layout width="fill parent"
    android:layout height="wrap content"
    android:onClick="onClickModify"/>
</LinearLayout>
```

（6）修改 LearningPreferencesActivity.java 文件，内容如下所示。

```
package com.selfteaching.learningpreferences;
```

```
import android.app.Activity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;

public class LearningPreferencesActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    public void onClickLoad(View view) {
        Intent i = new Intent("com.selfteaching.AppPrefActivity");
        startActivity(i);
    }

    public void onClickDisplay(View view) {

        SharedPreferences appPrefs =

getSharedPreferences("com.selfteaching.learningpreferences preferences",
MODE PRIVATE);


        DisplayText(appPrefs.getString("editTextPref", ""));
    }

    public void onClickModify(View view) {

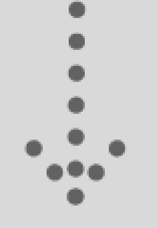
        SharedPreferences appPrefs =

getSharedPreferences("com.selfteaching.learningpreferences preferences",
MODE PRIVATE);

        SharedPreferences.Editor prefsEditor = appPrefs.edit();
        prefsEditor.putString("editTextPref",
                ((EditText) findViewById(R.id.txtString)).getText().
                toString());
        prefsEditor.commit();
    }

    private void DisplayText(String str) {
        Toast.makeText(getBaseContext(), str, Toast.LENGTH LONG).show();
    }

}
```





（7）在模拟器中运行该应用程序。单击“Load Preferences Screen”按钮显示 Preferences 界面，如图 8-1 所示。

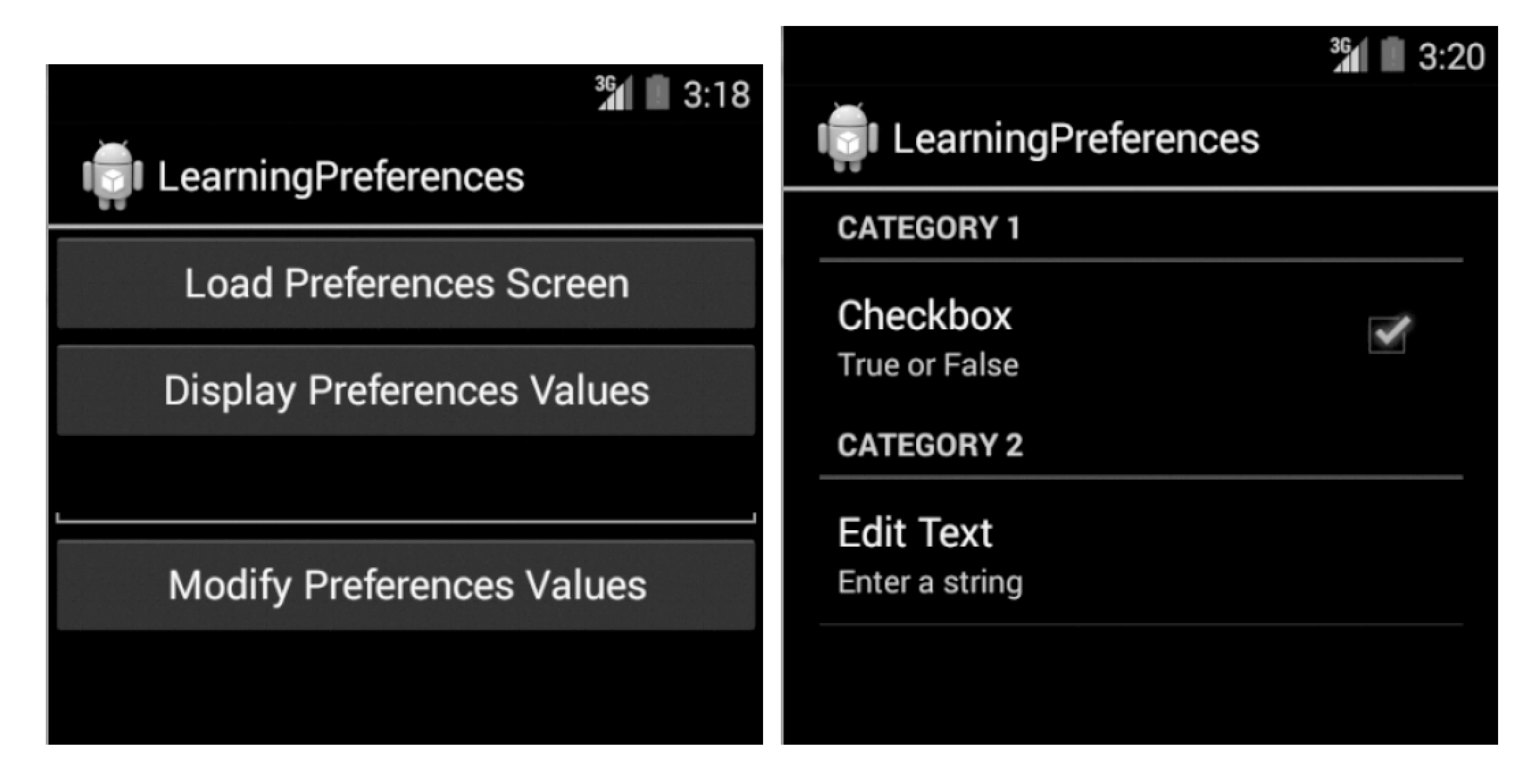


图 8-1　显示 Preferences 界面

（8）单击 Perferences 界面上的 Checkbox 复选框或 Edit Text 文本编辑项，使原始 Preferences 的值发生变化，然后离开 Preferences 界面。之后，一个新文件将会在 /data/data/com.selfteaching.learningpreferences/shared_prefs 文件夹中被创建出来，如图 8-2 所示。为了查看该文件，切换到 Eclipse 的 DDMS 视图，查看 File Explorer Tab，将会看见一个新的 XML 文件，名为 com.selfteaching.learningpreferences_preferences.xml。Preferences 界面中原始 Preferences 值的变化，都保存在该文件中。

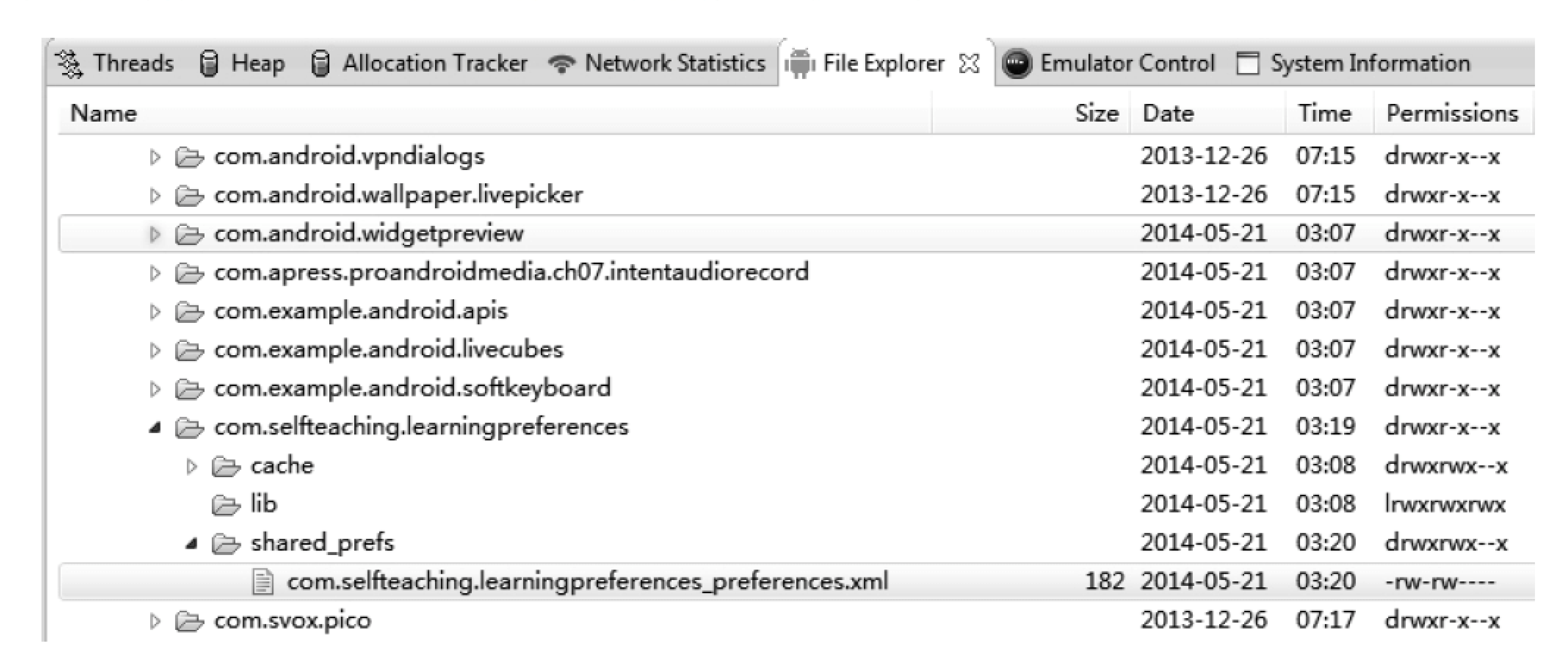


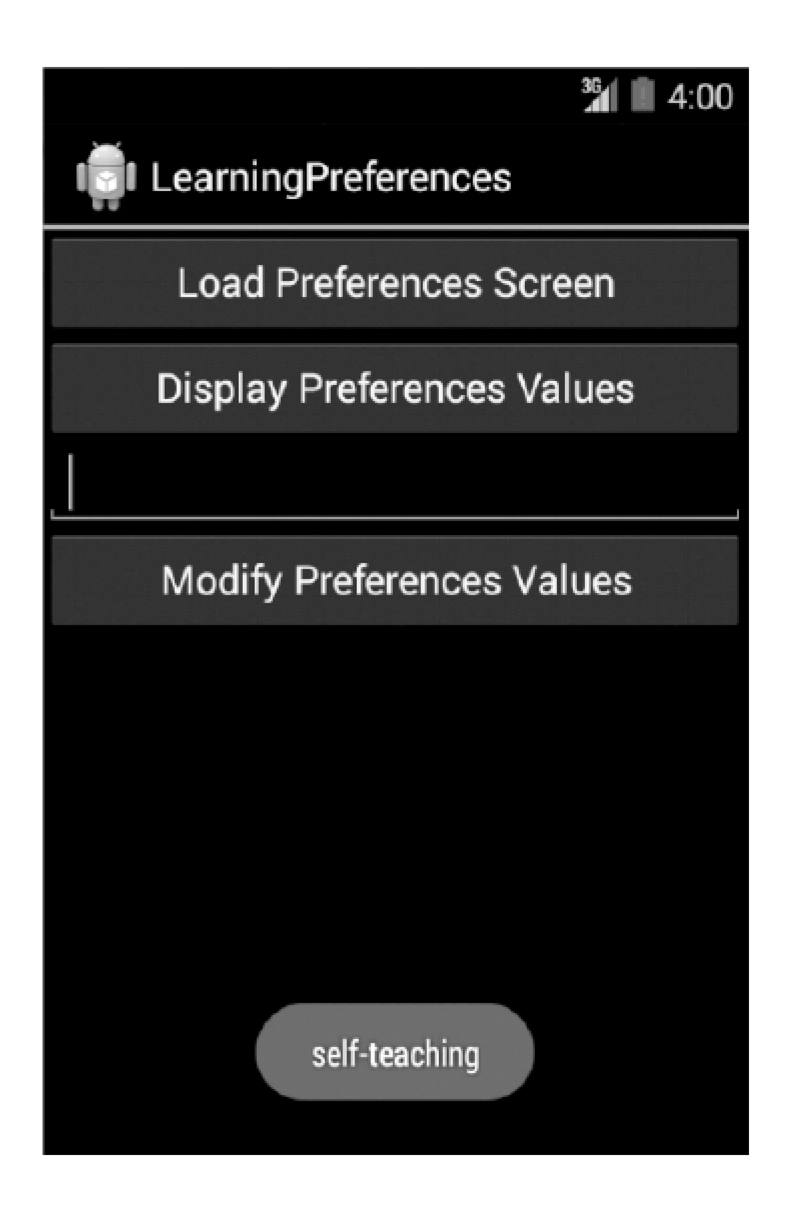
图 8-2　preferences xml 文件

（9）查看该文件的内容，如下所示。

```
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
```

```
    <string name="editTextPref">self-teaching</string>
    <boolean name="checkboxPref" value="true" />
    </map>
```

（10）单击“Display Preferences Values”按钮，显示如图 8-3 所示的界面。接着，在 EditText 中输入文本内容并且单击“Modify Preferences Values”按钮，如图 8-4 所示。





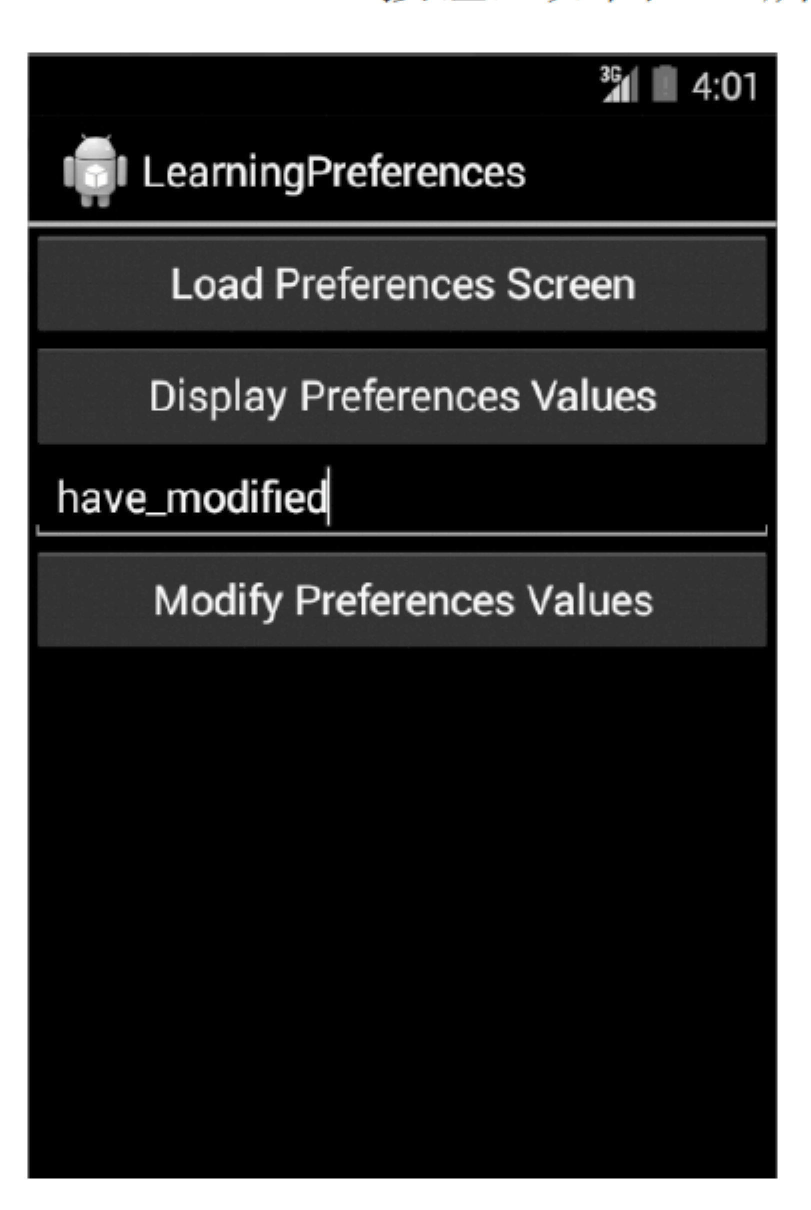


图 8-3　显示 Preferences Values　　　　图 8-4　修改 Preferences Values

（11）再次单击“Display Preferences Values”按钮，发现新修改的值被保存下来，如图 8-5 所示。

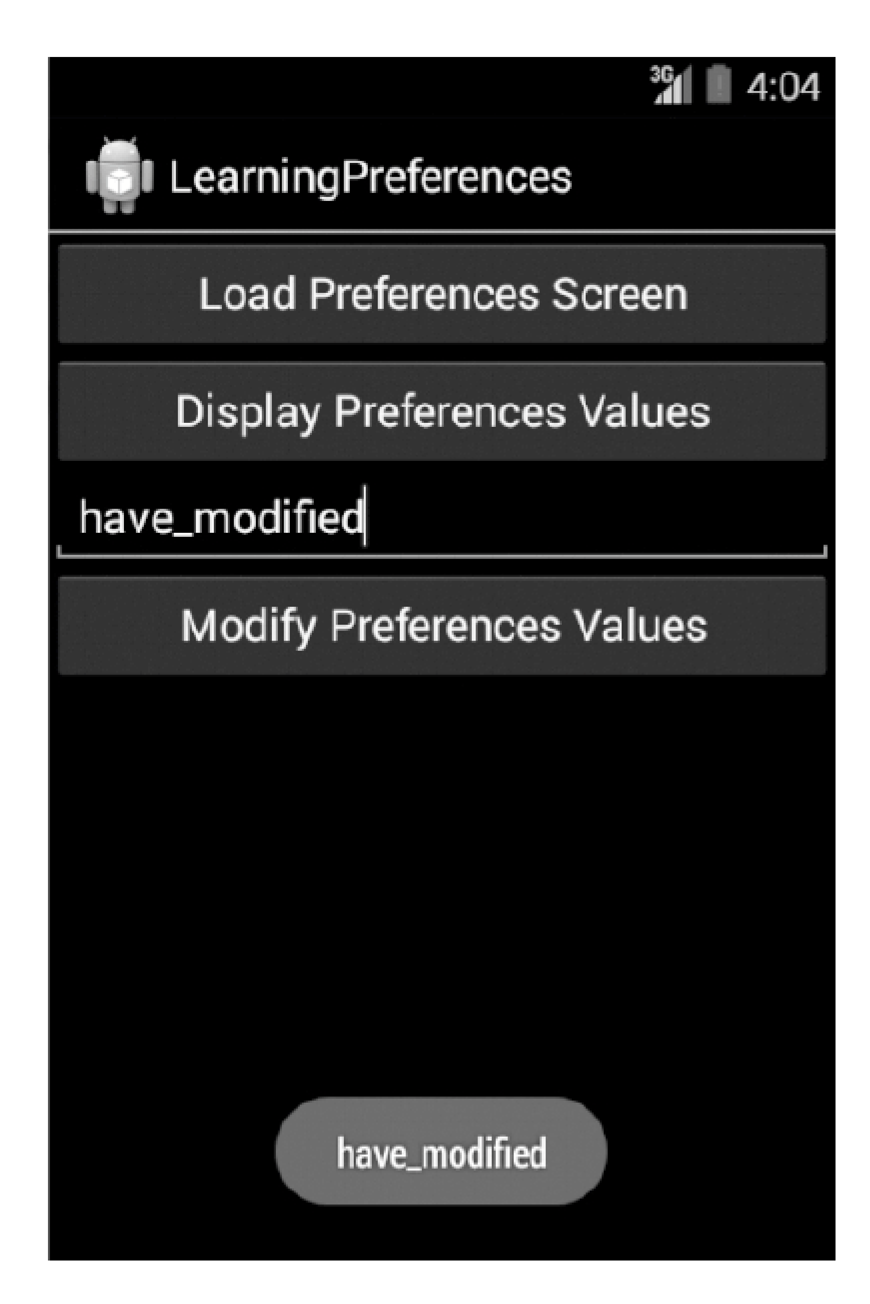


图 8-5　显示修改后的值

代码解释如下。

创建了名为 myappprefs.xml 的 xml 文件来指定应用程序要保存的 preferences 类型。

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen
xmlns:android="http://schemas.android.com/apk/res/android">
<PreferenceCategory android:title="Category 1">
   <CheckBoxPreference
      android:title="Checkbox"
      android:defaultValue="false"
      android:summary="True or False"
      android:key="checkboxPref" />
</PreferenceCategory>
<PreferenceCategory android:title="Category 2">
   <EditTextPreference
      android:summary="Enter a string"
      android:defaultValue="[Enter a string here]"
      android:title="Edit Text"
      android:key="editTextPref"
      />
   </PreferenceCategory>
</PreferenceScreen>
```

本例中创建了两个 preferences 类别用于分组 preferences 类型，第一个类别中包含一个 CheckBox Preference，key 为 checkboxPref，第二个类别中包含一个 EditText Preference，key 为 editTextPref。android:key 属性用于指定在代码中可以引用的 key，用它来设置或获取相应 preference 的值。

本例中创建了一个扩展自 PreferenceActivity 基类的派生类 AppPrefActivity，为了显示这些 preferences 以便用户编辑，任何再调用 addPreferencesFromResource()方法来加载包含 preferences 的 xml 文件。

```
public class AppPrefActivity extends PreferenceActivity {
    @Override
    public void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        //load preferences from the XML file
        addPreferencesFromResource(R.xml.myappprefs);
    }
}
```

通过使用一个 Intent 对象，启动该 Activity。

```
Intent i = new Intent("com.selfteaching.AppPrefActivity");
startActivity(i);
```

preferences 的所有改变将会自动地存储到应用程序的 shared_prefs 目录下的一个 xml 文件中。

为了显示 preferences 中的配置信息，在 onClickDisplay()方法中，首先使用 getSharedPreferences()方法获取 SharedPreferences 对象。本例中需要把上面自动创建的 xml

文件的文件名传给 getSharedPreferences()方法。该 xml 文件的命名方式是 <PackageName>_preferences。为了获取某个具体的 preference 的值，需使用 getString()方法，把 preference 的 key 传给该方法即可。

```
public void onClickDisplay(View view) {

    SharedPreferences appPrefs =

getSharedPreferences("com.selfteaching.learningpreferences preferences",
MODE PRIVATE);


    DisplayText(appPrefs.getString("editTextPref", ""));
    }
```

MODE_PRIVATE 常量表明该 preference 文件只能由创建它的应用程序打开。

在 onClickModify()方法中，首先需要创建 SharedPreferences.Editor 对象，该对象由 SharedPreferences 对象的 edit()方法创建。为了更改字符串 preference 的值，需使用 putString()方法。为了把变动内容保存到 xml 文件中，需调用 commit()方法。

```
public void onClickModify(View view) {

        SharedPreferences appPrefs =

getSharedPreferences("com.selfteaching.learningpreferences preferences",
MODE PRIVATE);

        SharedPreferences.Editor prefsEditor = appPrefs.edit();
        prefsEditor.putString("editTextPref",
                ((EditText) findViewById(R.id.txtString)).getText().
                toString());
        prefsEditor.commit();
    }
```

8.2 存储数据到文件

上节已经介绍过，SharedPreferences 对象允许用户存储名/值对数据，但是，有时用户也想使用文件系统来存储数据。在 Android 平台上，可以使用 java.io 包中的类来存储数据到文件中。本节介绍如何存储数据到内容存储和外部存储卡中。

案例：保存数据到内部存储中。

在 Android 应用程序中保存文件的第一种方式是把数据写到设备的内部存储中。本案例介绍如何把用户输入的文本内容保存到设备的内部存储中。

实现步骤如下。

（1）使用 Eclipse 创建一个新的 Android 项目，命名为 LearningFiles。

（2）修改 main.xml 文件，如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout width="fill parent"
    android:layout height="fill parent"
    android:orientation="vertical" >

<TextView
    android:layout width="fill parent"
    android:layout height="wrap content"
    android:text="Please enter some text" />

<EditText
    android:id="@+id/txtText1"
    android:layout width="fill parent"
    android:layout height="wrap content" />

<Button
    android:id="@+id/btnSave"
    android:text="Save"
    android:layout width="fill parent"
    android:layout height="wrap content"
    android:onClick="onClickSave" />

<Button
    android:id="@+id/btnLoad"
    android:text="Load"
    android:layout width="fill parent"
    android:layout height="wrap content"
    android:onClick="onClickLoad" />

</LinearLayout>
```

（3）修改 LearningFilesActivity.java 文件，如下所示。

```
package com.selfteaching.learningfiles;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;

import android.app.Activity;
import android.os.Bundle;
import android.os.Environment;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;
```

```
public class LearningFilesActivity extends Activity {
    EditText textBox;
    static final int READ_BLOCK_SIZE = 100;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        textBox = (EditText) findViewById(R.id.txtText1);

    }

    public void onClickSave(View view) {
        String str = textBox.getText().toString();
        try
        {

            FileOutputStream fOut =
                    openFileOutput("textfile.txt",
                            MODE_WORLD_READABLE);


            OutputStreamWriter osw = new
                    OutputStreamWriter(fOut);

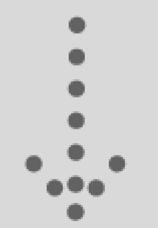
            //---write the string to the file---
            osw.write(str);
            osw.flush();
            osw.close();

            //---display file saved message---
            Toast.makeText(getBaseContext(),
                    "File saved successfully!",
                    Toast.LENGTH_SHORT).show();

            //---clears the EditText---
            textBox.setText("");
        }
        catch (IOException ioe)
        {
            ioe.printStackTrace();
        }
    }

    public void onClickLoad(View view) {
        try
        {

            FileInputStream fIn =
```



```
                openFileInput("textfile.txt");
        InputStreamReader isr = new
                InputStreamReader(fIn);


        char[] inputBuffer = new char[READ_BLOCK_SIZE];
        String s = "";

        int charRead;
        while ((charRead = isr.read(inputBuffer))>0)
        {
            //---convert the chars to a String---
            String readString =
                    String.copyValueOf(inputBuffer, 0,
                            charRead);
            s += readString;

            inputBuffer = new char[READ_BLOCK_SIZE];
        }
        //---set the EditText to the text that has been
        // read---
        textBox.setText(s);

        Toast.makeText(getBaseContext(),
                "File loaded successfully!",
                Toast.LENGTH_SHORT).show();
    }
    catch (IOException ioe) {
        ioe.printStackTrace();
    }

}

}
```

（4）在模拟器中运行该应用程序。在 EditText 文本编辑框中输入一段文本，如“hello files”，然后单击“保存”按钮，如图 8-6 所示。

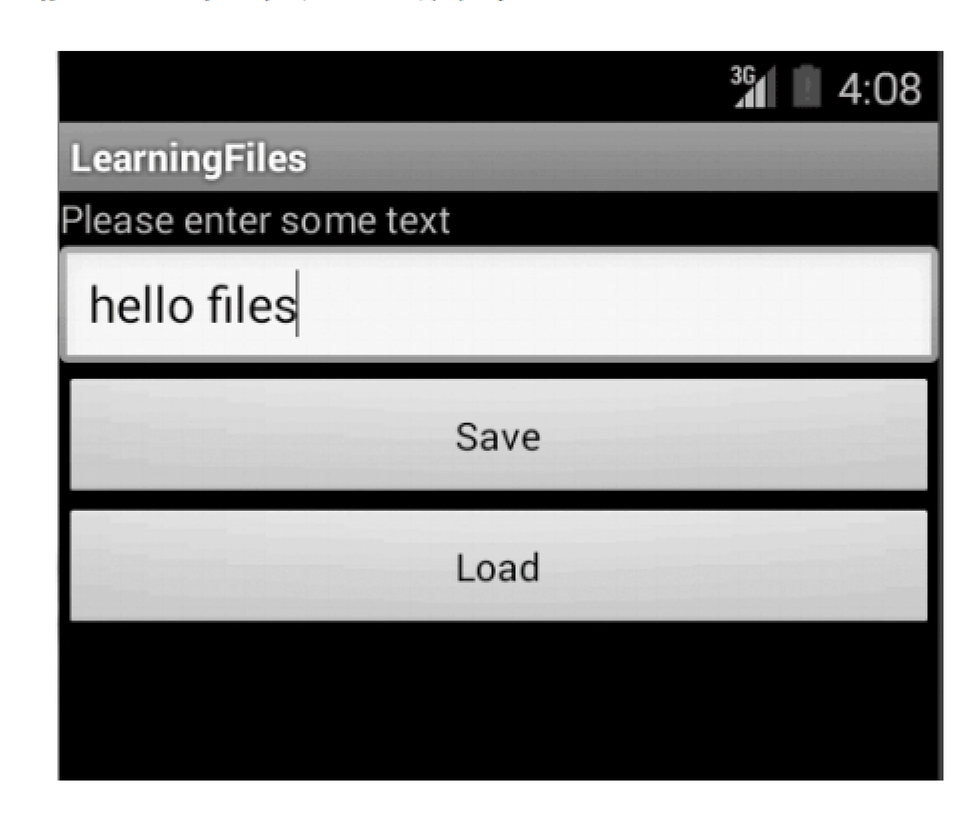


图 8-6　输入文本

（5）如果文件保存成功，会显示文件保存成功的提示信息，接下来 EditText 中的文本内容将会消失，如图 8-7 所示。

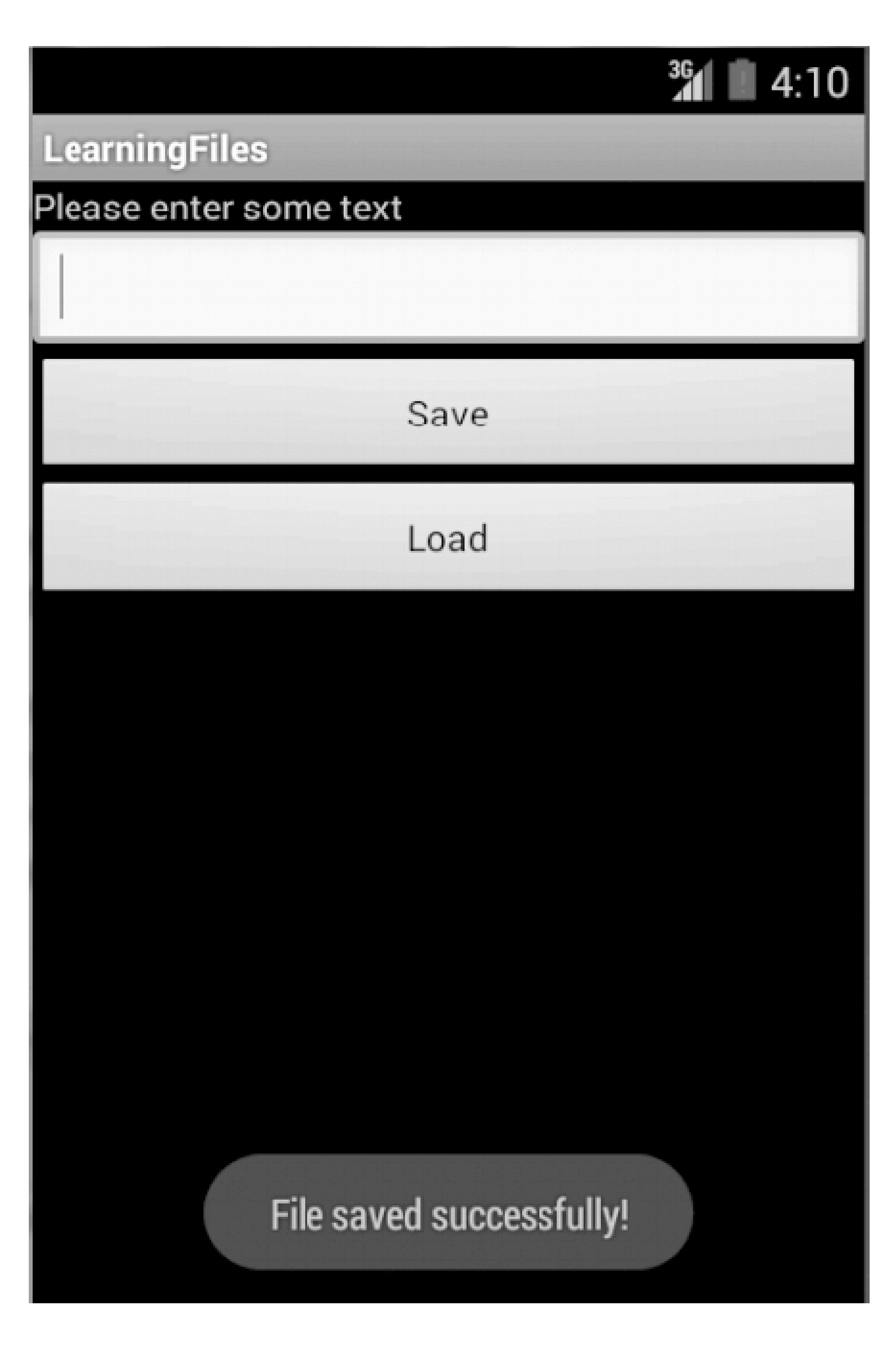


图 8-7　保存文件成功

（6）单击“加载”按钮，将会看见“hello files”字符串再次出现在 EditText 中，说明文件成功保存下来，如图 8-8 所示。

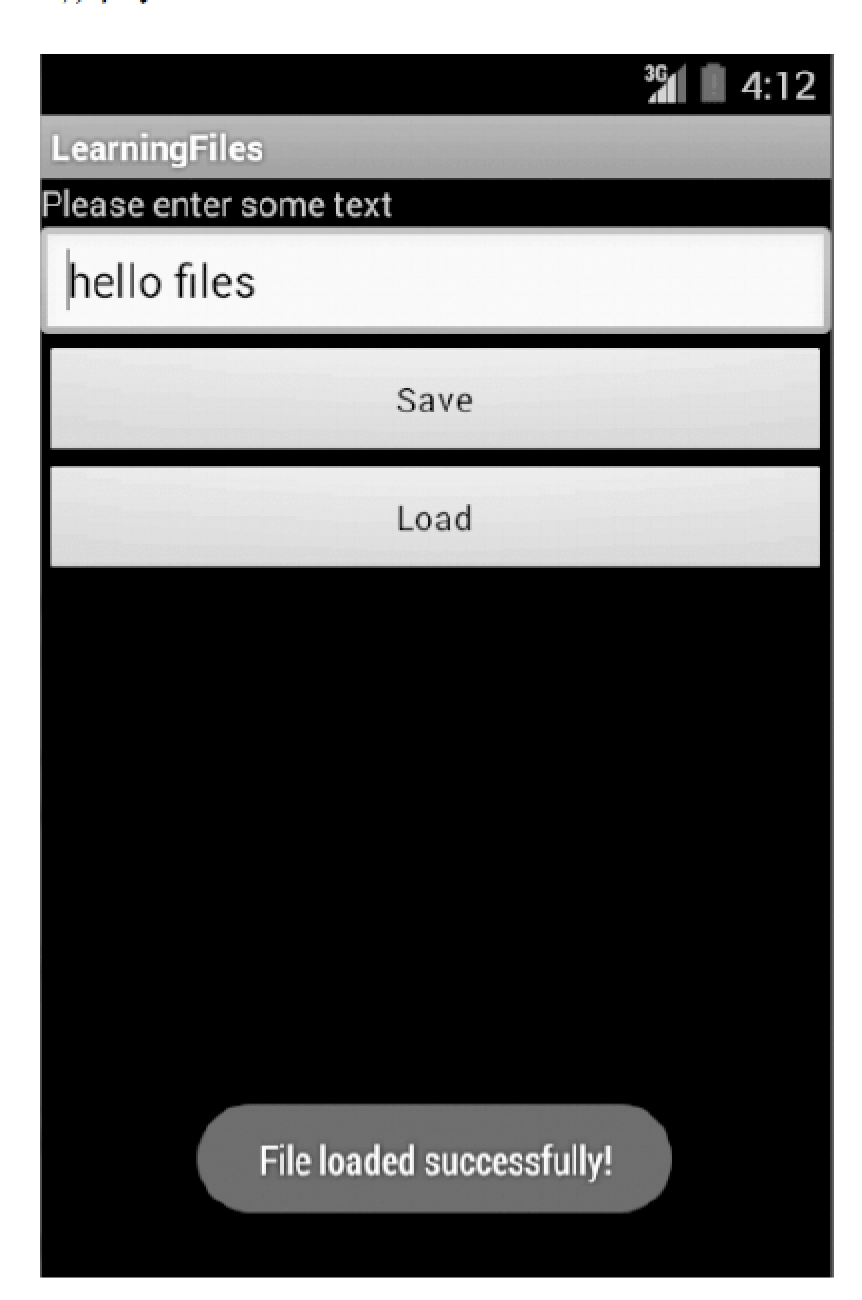
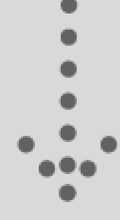


图 8-8　加载文件





代码解释如下。

保存本文到文件中，需要使用 FileOutputStream 类。openFileOutput()方法用于打开一个文件。本例中，使用 MODE_WORLD_READABLE 常量暗示该文件对所有应用来说都是可读的。

```
FileOutputStream fOut =
                   openFileOutput("textfile.txt",
                           MODE_WORLD_READABLE);
```

除了 MODE_WORLD_READABLE 常量外，还可选择 MODE_PRIVATE，表明该文件仅能由创建它的应用访问；MODE_APPEND，表明向文件中追加内容；MODE_WORLD_WRITEABLE，所有应用都可写入该文件。

创建 OutputStreamWriter 类的一个实例，把刚才创建的 FileOutputStream 对象 fOut 传给它，转换字符流到字节流。

```
            OutputStreamWriter osw = new
                    OutputStreamWriter(fOut);
```

然后使用 write()方法把字符串写入文件。为了确保所有字节都写入文件，再使用 flush()方法刷新下。最后，使用 close()方法关闭文件，写入完成。

```
            //---write the string to the file---
            osw.write(str);
            osw.flush();
            osw.close();
```

为了读入文件内容，使用 FileInputStream 和 InputStreamReader 类。

```
FileInputStream fIn =
                   openFileInput("textfile.txt");
            InputStreamReader isr = new
                    InputStreamReader(fIn);
```

因为事先不知道文件的大小，先把内容读入到一个 100 字符的缓冲块中，然后把读取的字符块复制到一个 String 对象中。

```
char[] inputBuffer = new char[READ BLOCK SIZE];
String s = "";

int charRead;
while ((charRead = isr.read(inputBuffer))>0)
{
    //---convert the chars to a String---
    String readString =
            String.copyValueOf(inputBuffer, 0,
                    charRead);
    s += readString;

    inputBuffer = new char[READ BLOCK SIZE];
}
```

在模拟器中运行应用程序，切换到 DDMS 视图查看应用是否创建了一个新文件，文件路径是/data/data/com.selfteaching.learningfiles/files，如图 8-9 所示。

Threads | Heap | Allocation Tracker | Network Statistics | File Explorer | Emulator Control | System Information

Name	Size	Date	Time	Permissions
▷ com.android.widgetpreview		2014-05-22	04:07	drwxr-x--x
▷ com.apress.proandroidmedia.ch07.intentaudiorecord		2014-05-22	04:07	drwxr-x--x
▷ com.example.android.apis		2014-05-22	04:07	drwxr-x--x
▷ com.example.android.livecubes		2014-05-22	04:07	drwxr-x--x
▷ com.example.android.softkeyboard		2014-05-22	04:07	drwxr-x--x
◢ com.selfteaching.learningfiles		2014-05-22	04:10	drwxr-x--x
▷ cache		2014-05-22	04:08	drwxrwx--x
◢ files		2014-05-22	04:10	drwxrwx--x
textfile.txt	11	2014-05-22	04:10	-rw-rw-r--
lib		2014-05-22	04:07	lrwxrwxrwx
▷ com.selfteaching.learningpreferences		2014-05-22	04:07	drwxr-x--x
▷ com.svox.pico		2013-12-26	07:17	drwxr-x--x
▷ jp.co.omronsoft.openwnn		2013-12-26	07:15	drwxr-x--x

图 8-9　DDMS 中查看新创建的文件

案例：保存数据到外部存储卡中。

因为外部存储卡通常有更大的容量，也更容易共享数据，因此，保存数据到外部存储卡上是一个更好的选择。

实现步骤如下。

（1）使用案例 1 的项目，修改 LearningFilesActivity.java 文件，如下所示。

```
public void onClickSave(View view) {
    String str = textBox.getText().toString();
    try
    {
        //---SD Card Storage---
        File sdCard = Environment.getExternalStorageDirectory();
        File directory = new File (sdCard.getAbsolutePath() +
            "/myfiles");
        directory.mkdirs();
        File file = new File(directory, "textfile.txt");
        FileOutputStream fOut = new FileOutputStream(file);

        /*
        FileOutputStream fOut = openFileOutput("textfile.txt",
          MODE_WORLD_READABLE);
        */

        OutputStreamWriter osw = new
        OutputStreamWriter(fOut);
        //---write the string to the file---
        osw.write(str);
        osw.flush();
        osw.close();
        //---display file saved message---
        Toast.makeText(getBaseContext(),"File saved successfully! ",
            Toast.LENGTH_SHORT).show();
```

```
            //---clears the EditText---
            textBox.setText("");
        }catch (IOException ioe){
            ioe.printStackTrace();
        }
    }

    public void onClickLoad(View view) {
        try
        {
          //---SD Storage---
          File sdCard = Environment.getExternalStorageDirectory();
          File directory = new File (sdCard.getAbsolutePath() +
               "/myfiles");
          File file = new File(directory, "textfile.txt");
          FileInputStream fIn = new FileInputStream(file);
          InputStreamReader isr = new InputStreamReader(fIn);

          /*
          FileInputStream fIn =
          openFileInput("textfile.txt");
          InputStreamReader isr = new InputStreamReader(fIn);
          */

          char[] inputBuffer = new char[READ_BLOCK_SIZE];
          String s = "";
          int charRead;
          while ((charRead = isr.read(inputBuffer))>0)
          {
            //---convert the chars to a String---
            String readString = String.copyValueOf(inputBuffer, 0,
                charRead);
            s += readString;
            inputBuffer = new char[READ_BLOCK_SIZE];
            }
            ... ...
    }
```

（2）修改 AndroidManifest.xml 文件，如下所示。

```
    <?xml version="1.0" encoding="utf-8"?>
    <manifest xmlns:android="http://schemas.android.com/apk/res/android"
              package="com.selfteaching.learningfiles"
              android:versionCode="1"
              android:versionName="1.0" >
              <uses-sdk android:minSdkVersion="14" />
              <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
              <application
              android:icon="@drawable/ic_launcher"
              android:label="@string/app_name" >
```

```
            <activity
            android:label="@string/app name"
            android:name=".LearningFilesActivity" >
            <intent-filter >
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
            </activity>
            </application>
</manifest>
```

代码解释如下。

getExternalStorageDirectory()方法可以返回外部存储卡的全路径。通常在真实设备上，它会返回“/sdcard”，而在模拟器上，会返回“/mnt/sdcard”。但是不要硬编码存储卡路径，因为设备商可能会更改存储卡的路径。始终使用 getExternalStorageDirectory()方法获取外部存储卡的全路径。

接着，使用 File 对象的 mkdirs()方法在存储卡上创建一个目录用于保存文件。

```
File directory = new File (sdCard.getAbsolutePath() +
     “/myfiles");
directory.mkdirs();
```

为了可以写数据到外部存储卡上，需要在 AndroidManifest.xml 文件中添加 WRITE_EXTERNAL_STORAGE 权限。

在模拟器上运行应用程序，会在“/mnt/sdcard/myfiles”文件夹下看见创建的文件。

8.3　使用数据库存储数据

到目前为止，前面介绍的技术只对存储简单的数据有效。如需保存关系型数据，使用数据库会更加高效。例如，如果用户想保存学校里所有学生的测验成绩，使用数据库保存它们会更加高效，因为用户能使用数据库查询方便地检索特定学生的成绩。而且，使用数据库能很好地保证数据完整性。Android 平台使用 SQLite 数据库系统。为一个应用程序创建的数据库只能被它自己访问，其他应用程序没有访问权限。本节主要使读者学会如何在应用程序中动态地编写创建、增加、删除、修改、查询数据库的程序。在 Android 平台中，应用程序创建的 SQLite 数据库总是被存放在/data/data/<package_name>/databases 文件夹下。

创建一个数据库帮助类 DBAdapter 来包装复杂的数据库操作是一个良好的编码习惯，它可以隐藏具体的数据库操作细节，使用户只需关注业务逻辑的实现。

案例：创建数据库帮助类 DBAdapter。

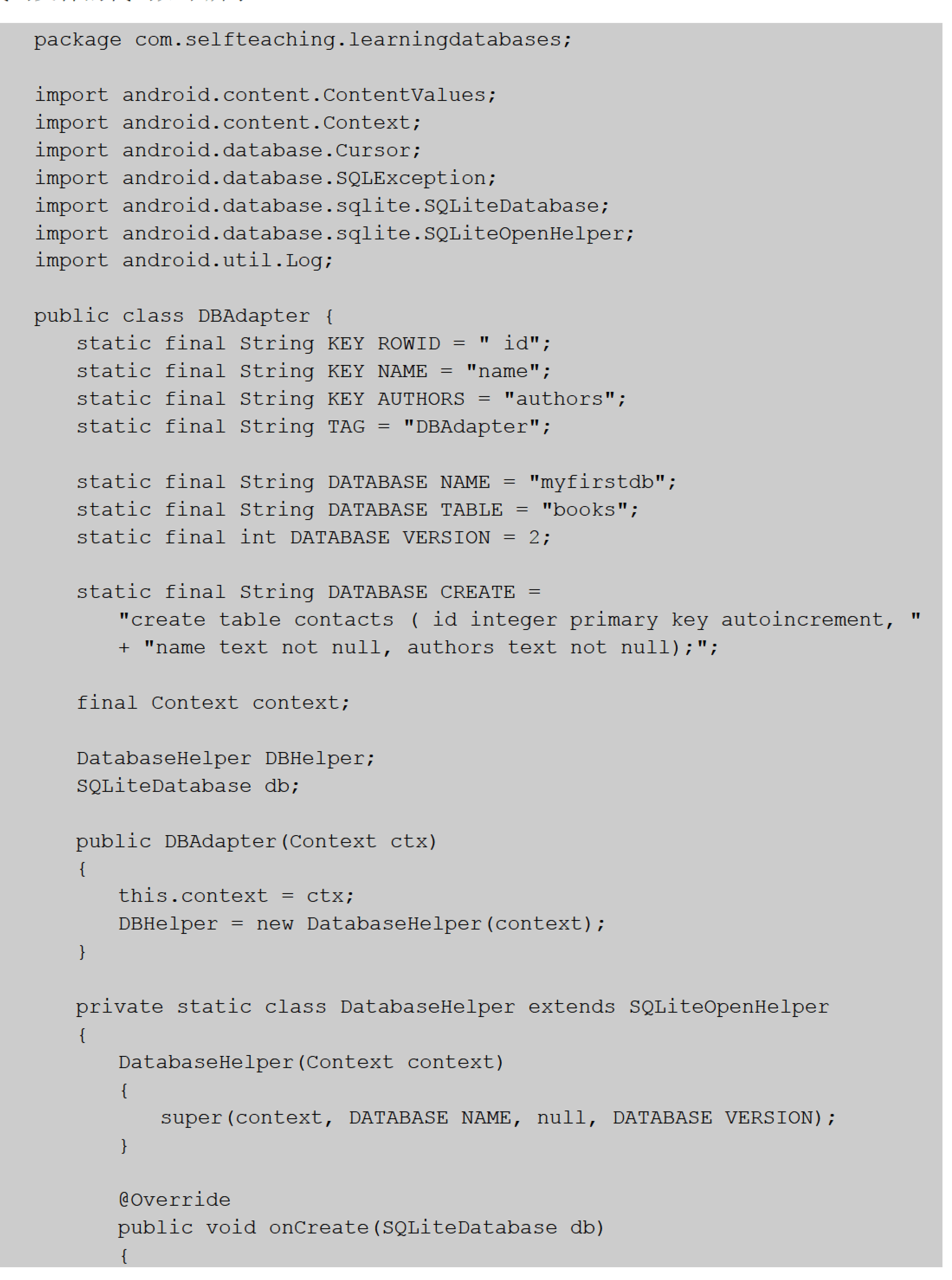
本例将创建一个名为 myfirstdb 的数据库，其中包含一张名为 books 的数据表，该表有 3 列：_id、name 和 authors。通过该案例，读者可以学会如何创建、打开、关闭及操作数据库。



实现步骤如下。

（1）使用 Eclipse 创建一个新的 Android 项目，命名为 LearningDatabases。

（2）在 src 文件夹中，添加一个新的 Java 源代码文件，命名为 DBAdapter，该 Java 源代码文件的代码如下所示。



```
package com.selfteaching.learningdatabases;

import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.util.Log;

public class DBAdapter {
    static final String KEY ROWID = " id";
    static final String KEY NAME = "name";
    static final String KEY AUTHORS = "authors";
    static final String TAG = "DBAdapter";

    static final String DATABASE NAME = "myfirstdb";
    static final String DATABASE TABLE = "books";
    static final int DATABASE VERSION = 2;

    static final String DATABASE CREATE =
        "create table contacts ( id integer primary key autoincrement, "
        + "name text not null, authors text not null);";

    final Context context;

    DatabaseHelper DBHelper;
    SQLiteDatabase db;

    public DBAdapter(Context ctx)
    {
        this.context = ctx;
        DBHelper = new DatabaseHelper(context);
    }

    private static class DatabaseHelper extends SQLiteOpenHelper
    {
        DatabaseHelper(Context context)
        {
            super(context, DATABASE NAME, null, DATABASE VERSION);
        }

        @Override
        public void onCreate(SQLiteDatabase db)
        {
```

```
            try {
                db.execSQL(DATABASE_CREATE);
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }

        @Override
        public void onUpgrade(SQLiteDatabase db, int oldVersion, int
        newVersion)
        {
            Log.w(TAG, "Upgrading database from version " + oldVersion + "
            to "
                    + newVersion + ", which will destroy all old data");
            db.execSQL("DROP TABLE IF EXISTS books");
            onCreate(db);
        }
    }

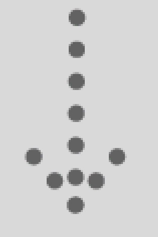
    //---opens the database---
    public DBAdapter open() throws SQLException
    {
        db = DBHelper.getWritableDatabase();
        return this;
    }

    //---closes the database---
    public void close()
    {
        DBHelper.close();
    }

    //---insert a book into the database---
    public long insertBook(String name, String authors)
    {
        ContentValues initialValues = new ContentValues();
        initialValues.put(KEY_NAME, name);
        initialValues.put(KEY_AUTHORS, authors);
        return db.insert(DATABASE_TABLE, null, initialValues);
    }

    //---deletes a particular book---
    public boolean deleteBook(long rowId)
    {
        return db.delete(DATABASE_TABLE, KEY_ROWID + "=" + rowId, null) > 0;
    }

    //---retrieves all the books---
    public Cursor getAllBooks()
    {
        return db.query(DATABASE_TABLE, new String[] {KEY_ROWID, KEY_NAME,
```



```
                KEY AUTHORS}, null, null, null, null, null);
    }

    //---retrieves a particular book---
    public Cursor getBook(long rowId) throws SQLException
    {
        Cursor mCursor =
                db.query(true, DATABASE TABLE, new String[] {KEY ROWID,
                KEY NAME, KEY AUTHORS}, KEY ROWID + "=" + rowId, null,
                null, null, null, null);
        if (mCursor != null) {
            mCursor.moveToFirst();
        }
        return mCursor;
    }

    //---updates a book---
    public boolean updateBook(long rowId, String name, String authors)
    {
        ContentValues args = new ContentValues();
        args.put(KEY NAME, name);
        args.put(KEY AUTHORS, authors);
        return db.update(DATABASE TABLE, args, KEY ROWID + "=" + rowId, null)
        > 0;
    }

}
```

代码解释如下。

首先，创建了几个常量，表示数据库名，数据表名和数据表的字段名。

```
static final String KEY ROWID = " id";
static final String KEY NAME = "name";
static final String KEY AUTHORS = "authors";
static final String TAG = "DBAdapter";

static final String DATABASE NAME = "myfirstdb";
static final String DATABASE TABLE = "books";
static final int DATABASE VERSION = 2;

static final String DATABASE CREATE =
    "create table contacts ( id integer primary key autoincrement, "
    + "name text not null, authors text not null);";
```

DATABASE_CREATE 常量包含创建数据表 books 的 SQL 语句。

在 DBAdapter 类中，我们创建了一个扩展自 SQLiteOpenHelper 类的私有类，负责管理数据库的创建和版本管理。具体地，我们重载了 onCreate()和 onUpgrade()方法。

```
private static class DatabaseHelper extends SQLiteOpenHelper
    {
        DatabaseHelper(Context context)
```

```
        {
            super(context, DATABASE_NAME, null, DATABASE_VERSION);
        }

        @Override
        public void onCreate(SQLiteDatabase db)
        {
            try {
                db.execSQL(DATABASE_CREATE);
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }

        @Override
        public void onUpgrade(SQLiteDatabase db, int oldVersion, int
        newVersion)
        {
            Log.w(TAG, "Upgrading database from version " + oldVersion + "
            to "
                    + newVersion + ", which will destroy all old data");
            db.execSQL("DROP TABLE IF EXISTS books");
            onCreate(db);
        }
    }
```

如果数据库不存在，onCreate()方法创建一个新的数据库 myfirstdb。当数据库需要更新时，onUpgrade()方法被调用。通过检测在 DATABASE_VERSION 常量中的值来确定是否需要更新数据库。对于 onUpgrade()方法的具体实现，简单地删除数据表 books，并且重新创建它。

然后，定义打开和关闭数据库的方法，以及增加、删除和修改数据表中记录的方法。

```
    //---opens the database---
    public DBAdapter open() throws SQLException
    {
        db = DBHelper.getWritableDatabase();
        return this;
    }

    //---closes the database---
    public void close()
    {
        DBHelper.close();
    }

    //---insert a book into the database---
    public long insertBook(String name, String authors)
    {
        ContentValues initialValues = new ContentValues();
        initialValues.put(KEY_NAME, name);
```

```
            initialValues.put(KEY AUTHORS, authors);
            return db.insert(DATABASE TABLE, null, initialValues);
        }

        //---deletes a particular book---
        public boolean deleteBook(long rowId)
        {
            return db.delete(DATABASE TABLE, KEY ROWID + "=" + rowId, null) > 0;
        }

        //---retrieves all the books---
        public Cursor getAllBooks()
        {
            return db.query(DATABASE TABLE, new String[] {KEY ROWID, KEY NAME,
                    KEY AUTHORS}, null, null, null, null, null);
        }

        //---retrieves a particular book---
        public Cursor getBook(long rowId) throws SQLException
        {
            Cursor mCursor =
                    db.query(true, DATABASE TABLE, new String[] {KEY ROWID,
                    KEY NAME, KEY AUTHORS}, KEY ROWID + "=" + rowId, null,
                    null, null, null, null);
            if (mCursor != null) {
                mCursor.moveToFirst();
            }
            return mCursor;
        }

        //---updates a book---
        public boolean updateBook(long rowId, String name, String authors)
        {
            ContentValues args = new ContentValues();
            args.put(KEY NAME, name);
            args.put(KEY AUTHORS, authors);
            return db.update(DATABASE TABLE, args, KEY ROWID + "=" + rowId, null)
            > 0;
        }
```

Android 使用 Cursor 类作为数据库查询的返回值。可以把 Cursor 看作是指向数据库查询出的结果集的指针。使用 Cursor 可使 Android 更加高效地管理记录和字段。

ContentValues 对象可以存储名/值对，其 put()方法可以插入具有不同数据类型值的名/值对。

使用 DBAdapter 类在应用程序中创建数据库时，首先要创建 DBAdapter 类的示例。

```
    public DBAdapter(Context ctx)
        {
            this.context = ctx;
            DBHelper = new DatabaseHelper(context);
```

```
        }
```

DBAdapter 类的构造函数随后创建 DatabaseHelper 类的示例，用于真正地创建一个数据库。

```
DatabaseHelper(Context context)
        {
            super(context, DATABASE NAME, null, DATABASE VERSION);
        }
```

案例：添加新书到数据表 books 中。

实现步骤如下。

使用案例 1 的项目，修改 LearningDatabasesActivity.java 文件，如下所示。

```
package com.selfteaching.learningdatabases;

import android.app.Activity;
import android.os.Bundle;

public class LearningDatabasesActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        DBAdapter db = new DBAdapter(this);
        //---add a book---
        db.open();
        long id = db.insertBook("Learning Android", "Jim");
        id = db.insertBook("Learning Python", "Tom");
        db.close();
    }
}
```

代码解释如下。

本例中，首先创建 DBAdapter 类的一个实例，然后调用它的 insertBook()方法插入一条记录，即向数据表 books 中插入一本书的记录，返回插入记录的 ID。如果插入时有错误发生，则返回–1。

```
DBAdapter db = new DBAdapter(this);
```

切换到 DDMS 视图，查看 File Explorer Tab，会发现此时在 databases 文件夹下出现了 myfirstdb 文件，说明 myfirstdb 数据库已经建立。

案例：获取数据表 books 中的所有书本记录。

实现步骤如下。

使用案例 2 的项目，修改 LearningDatabasesActivity.java 文件，如下所示。

```
package com.selfteaching.learningdatabases;
```




```
import android.app.Activity;
import android.os.Bundle;

public class LearningDatabasesActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        DBAdapter db = new DBAdapter(this);

        /*
        //---add two books---
        db.open();
        long id = db.insertBook("Learning Android", "Jim");
        id = db.insertBook("Learning Python", "Tom");
        db.close();
        */

        //--get all contacts---
        db.open();
        Cursor c = db.getAllContacts();
        if (c.moveToFirst())
        {
            do {
                DisplayBook(c);
            } while (c.moveToNext());
        }
        db.close();

    }

    public void DisplayBook(Cursor c)
    {
        Toast.makeText(this,
                "id: " + c.getString(0) + "\n" +
                "Name: " + c.getString(1) + "\n" +
                "Authors:  " + c.getString(2),
                Toast.LENGTH LONG).show();
    }
}
```

代码解释如下。

DBAdapter 类的 getAllBooks()方法获取数据库中的所有书本记录。返回结果是一个 Cursor 对象。为了显示所有书本记录，必须首先调用 Cursor 对象的 moveToFirst()方法，目的是移动到第一条记录。如果成功，表明返回结果至少有一条记录，然后调用 DisplayBook()方法显示书本详情。调用 Cursor 对象的 moveToNext()方法可以移到下一条记录。

案例：获取数据表 books 中的某一条书本记录。

实现步骤如下。

使用案例 3 的项目，修改 LearningDatabasesActivity.java 文件，如下所示。

```
package com.selfteaching.learningdatabases;

import android.app.Activity;
import android.os.Bundle;

public class LearningDatabasesActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

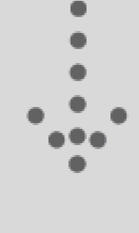
        DBAdapter db = new DBAdapter(this);

        /*
        //---add two books---
        db.open();
        long id = db.insertBook("Learning Android", "Jim");
        id = db.insertBook("Learning Python", "Tom");
        db.close();
        */

        /*
        //--get all books---
        db.open();
        Cursor c = db.getAllContacts();
        if (c.moveToFirst())
        {
            do {
                DisplayBook(c);
            } while (c.moveToNext());
        }
        db.close();
        */

        //---get a book---
        db.open();
        Cursor c = db.getBook(2);
        if (c.moveToFirst())
            DisplayContact(c);
        else
           Toast.makeText(this,"No book found", Toast.LENGTH_LONG).show();
        db.close();
    }

    public void DisplayBook(Cursor c)
```



```
        {
            Toast.makeText(this,
                    "id: " + c.getString(0) + "\n" +
                    "Name: " + c.getString(1) + "\n" +
                    "Authors:  " + c.getString(2),
                    Toast.LENGTH LONG).show();
        }
    }
```

在模拟器中运行该应用程序。第二个书本记录的详细信息将会显示在界面上。

代码解释如下。

DBAdapter 类的 getBook()方法返回由参数 id 指定的单条书本记录。本例中传给 getBook()方法参数 id 值 2，表明想要获取第二条书本记录。

```
Cursor c = db.getBook(2);
```

返回结果是一个 Cursor 对象。如果一条记录被返回，则用 DisplayBook()方法显示该书本的详情，否则，显示没有任何书本的提示信息。

案例：更新数据表 books 中的某一条书本记录的信息。

实现步骤如下。

使用案例 4 的项目，修改 LearningDatabasesActivity.java 文件，如下所示。

```
package com.selfteaching.learningdatabases;

import android.app.Activity;
import android.os.Bundle;

public class LearningDatabasesActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        DBAdapter db = new DBAdapter(this);

        /*
        //---add two books---
        db.open();
        long id = db.insertBook("Learning Android", "Jim");
        id = db.insertBook("Learning Python", "Tom");
        db.close();
        */

        /*
        //--get all books---
        db.open();
        Cursor c = db.getAllContacts();
        if (c.moveToFirst())
```

```
            {
                do {
                    DisplayBook(c);
                } while (c.moveToNext());
            }
            db.close();
            */

            /*
            //---get a book---
            db.open();
            Cursor c = db.getBook(2);
            if (c.moveToFirst())
                DisplayContact(c);
            else
                Toast.makeText(this, "No book found", Toast.LENGTH LONG).show();
            db.close();
            */

            //---update contact---
            db.open();
            if (db.updateBook(1, "Learning Android 2", "Smith"))
                Toast.makeText(this, "Update successful.", Toast.LENGTH LONG).
                show();
            else
                Toast.makeText(this, "Update failed.", Toast.LENGTH LONG).
                show();
            db.close();

        }

        public void DisplayBook(Cursor c)
        {
            Toast.makeText(this,
                    "id: " + c.getString(0) + "\n" +
                    "Name: " + c.getString(1) + "\n" +
                    "Authors:  " + c.getString(2),
                    Toast.LENGTH LONG).show();
        }
    }
```

在模拟器中运行该应用程序。更新后会显示更新是否成功的提示信息。

代码解释如下。

DBAdapter 类的 updateBook()方法负责更新书本的详情，参数是需要更新的某本书本的 id 值，该方法返回布尔值，表明是否更新成功。

案例：删除数据表 books 中的某一条书本记录。

实现步骤如下。

使用案例 5 的项目，修改 LearningDatabasesActivity.java 文件，如下所示。

```
package com.selfteaching.learningdatabases;

import android.app.Activity;
import android.os.Bundle;

public class LearningDatabasesActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        DBAdapter db = new DBAdapter(this);

        /*
        //---add two books---
        db.open();
        long id = db.insertBook("Learning Android", "Jim");
        id = db.insertBook("Learning Python", "Tom");
        db.close();
        */

        /*
        //--get all books---
        db.open();
        Cursor c = db.getAllContacts();
        if (c.moveToFirst())
        {
            do {
                DisplayBook(c);
            } while (c.moveToNext());
        }
        db.close();
        */

        /*
        //---get a book---
        db.open();
        Cursor c = db.getBook(2);
        if (c.moveToFirst())
            DisplayContact(c);
        else
           Toast.makeText(this, "No book found", Toast.LENGTH LONG).show();
        db.close();
        */

       /*
        //---update contact---
       db.open();
       if (db.updateBook(1, "Learning Android 2", "Smith"))
           Toast.makeText(this, "Update successful.", Toast.LENGTH_LONG).
```

```
            show();
        else
            Toast.makeText(this, "Update failed.", Toast.LENGTH LONG).
            show();
        db.close();
        */

    //---delete a contact---
        db.open();
        if (db.deleteBook(1))
            Toast.makeText(this, "Delete successful.", Toast.LENGTH LONG).
           show();
        else
            Toast.makeText(this, "Delete failed.", Toast.LENGTH LONG).
           show();
        db.close();

    }

    public void DisplayBook(Cursor c)
    {
        Toast.makeText(this,
               "id: " + c.getString(0) + "\n" +
               "Name: " + c.getString(1) + "\n" +
               "Authors:  " + c.getString(2),
               Toast.LENGTH LONG).show();
    }
}
```

在模拟器中运行该应用程序，删除后会显示删除是否成功的提示信息。

代码解释如下。

DBAdapter 类的 deleteBook()方法负责删除书本记录，参数是需要删除的某本书本的 id 值。该方法返回布尔值，表明是否删除成功。

8.4　本章小结

本章主要介绍了 Android 平台的三种数据存储的相关知识和用法。首先，讲解了 Android 平台提供的 SharedPreferences 对象，怎样用它来保存简单的应用程序数据；然后，对于非键值对数据，讲解了如何使用文件来存储文本内容；最后，讲解如何使用关系型数据库来保存关系型数据，以便更好地进行检索。

第 9 章　多媒体开发和电话 API

本章主要介绍 Android 平台的多媒体开发和短信电话方面的基础知识，以及学习如何在 Android 应用程序中播放音视频、发送短信和拨打电话。多媒体主要包括音频和视频，为方便起见，本章将分开介绍音频和视频的录制与播放相关的功能，此外，本章还将讲解如何发送接收短信和拨打电话的功能。

9.1　多媒体开发

当前，智能手机的功能十分强大，除了常规的拨打电话外，还可以浏览互联网，听音乐，看视频，甚至观看在线电影。本节将带你学习如何使用 Android 播放音频文件，观看视频，以及录制音频和视频。Android 平台提供强大的多媒体播放功能，它支持相当广泛的音频和视频格式。本节将首先介绍 Android 平台支持的常见音视频格式，然后介绍播放音频和视频的方法，最后讲解如何录制音频和视频。

9.1.1　常见的多媒体格式

Android 支持多种音频格式和编解码器，下面介绍几种常见的音频格式。

（1）AMR：自适应多速率编解码器，包括 AMR 窄带 AMR-NB 和 AMR 宽带 AMR-WB，文件扩展名是.3gp（audio/3gpp）或.amr（audio/amr）。AMR 是 3GPP 使用的基本音频编解码标准。AMR 主要应用于手机上的语音通话应用，并得到手机厂商的广泛支持，该编码标准适应于简单的语音编码，不适用处理更复杂的音频数据，如，音乐等。

（2）AAC：全称是 Advanced Audio Coding。一种专为声音数据设计的文件压缩格式，与 MP3 不同，它采用了全新的算法进行编码，更加高效，具有更高的"性价比"。利用 AAC 格式，可使人感觉声音质量没有明显降低的前提下，更加小巧。Android 除了支持 AAC 外，还支持新添加到 AAC 规范中的高效 AAC（High Efficiency AAC）格式。

（3）MP3：是一种音频压缩技术，其全称是动态影像专家压缩标准音频层面 3（Moving Picture Experts Group Audio Layer III），简称为 MP3。它被设计用来大幅度地降低音频数据量。利用 MPEG Audio Layer 3 的技术，将音乐以 1:10 甚至 1:12 的压缩率，压缩成容量较小的文件，而对于大多数用户来说重放的音质与最初的不压缩音频相比没有明显的下降。它是在 1991 年由位于德国埃尔朗根的研究组织 Fraunhofer-Gesellschaft 的一组工程师发明和标准化的。用 MP3 形式存储的音乐就叫作 MP3 音乐，能播放 MP3 音乐的机器就叫作 MP3 播放器。MP3 是目前互联网上使用最广泛的音频编解码器之一。

（4）Ogg：全称是 Ogg Vorbis，是一种新的音频压缩格式，类似于 MP3 的音乐格式。

但有一点不同的是，它是完全免费、开放和没有专利限制的。Ogg Vorbis 有一个特点是支持多声道。Vorbis 是这种音频压缩机制的名字，而 Ogg 则是一个计划的名字，该计划意图设计一个完全开放性的多媒体系统。Ogg Vorbis 文件的扩展名是.ogg，这种文件的设计格式是非常先进的。创建的 Ogg 文件可以在未来的任何播放器上播放，因此，这种文件格式可以不断地进行大小和音质的改良，而不影响旧有的编码器或播放器。

除了音频外，Android 还支持多种视频格式和编解码器，并且支持的类型还在不断增加，下面介绍几种常见的视频格式：

（1）H.263：由 ITU-T 制定的低码率视频编码标准，主要用于低延迟和低比特率的视频会议应用中。MPEG4（.mp4）和 3GP（.3gp）文件中都支持 H.263 编码的视频。

（2）H.264：也称为 MPEG-4 part 10 或 AVC（高级视频编码，Advanced Video Coding）。它是视频编码器的最新标准，在软件和硬件方面有广泛支持。H.264 被 Silverlight、Flash、iphone/ipod 以及蓝光设备等所支持。Android 以 MPEG-4 容器格式（.mp4）支持 H.264 编码的视频。

9.1.2 播放音频

本节主要介绍如何使用 MediaPlayer 类来播放音频。MediaPlayer 类是 Android SDK 提供的播放音频和视频的功能类，这里仅使用其音频播放功能，使用它可以建立功能更加完善的音频播放应用。

MediaPlayer 播放音频的最简单情况是播放与应用程序本身一起打包的音频文件。音频文件放置在应用程序的原始资源中。具体操作是在项目的 res 文件夹中创建一个新文件夹，命名为 raw，把音频文件放置入该 raw 文件夹中，ADT 将自动更新 R.java 文件（位于 gen 文件夹中），为该音频文件生成资源 ID，使用 R.raw.file_name_without_extension 语法访问该音频文件。

播放与应用程序一起打包的音频文件非常简单。使用 MediaPlayer 类的静态方法 create 实例化一个 MediaPlayer 对象，传入上下文 this 以及音频文件的资源 ID。

```
MediaPlayer mediaPlayer =
MediaPlayer.create(this, R.raw.audio_file_name_without_extension);
```

由于调用 MediaPlayer 的静态方法 create 创建 MediaPlayer 对象成功后，系统自动调用 prepare()方法，不再需要手动调用，MediaPlayer 对象已经处于 Prepared 状态，因此，只需调用 MediaPlayer 对象的 start()方法即可播放该音频文件。

```
mediaPlayer.start();
```

MediaPlayer 类内部维护一个状态机来管理播放音频和视频中的各种状态，该状态机如图 9-1 所示（摘自 Android API 参考手册），它描述了音频和视频播放过程中的各种状态和每个状态下可以调用的方法。

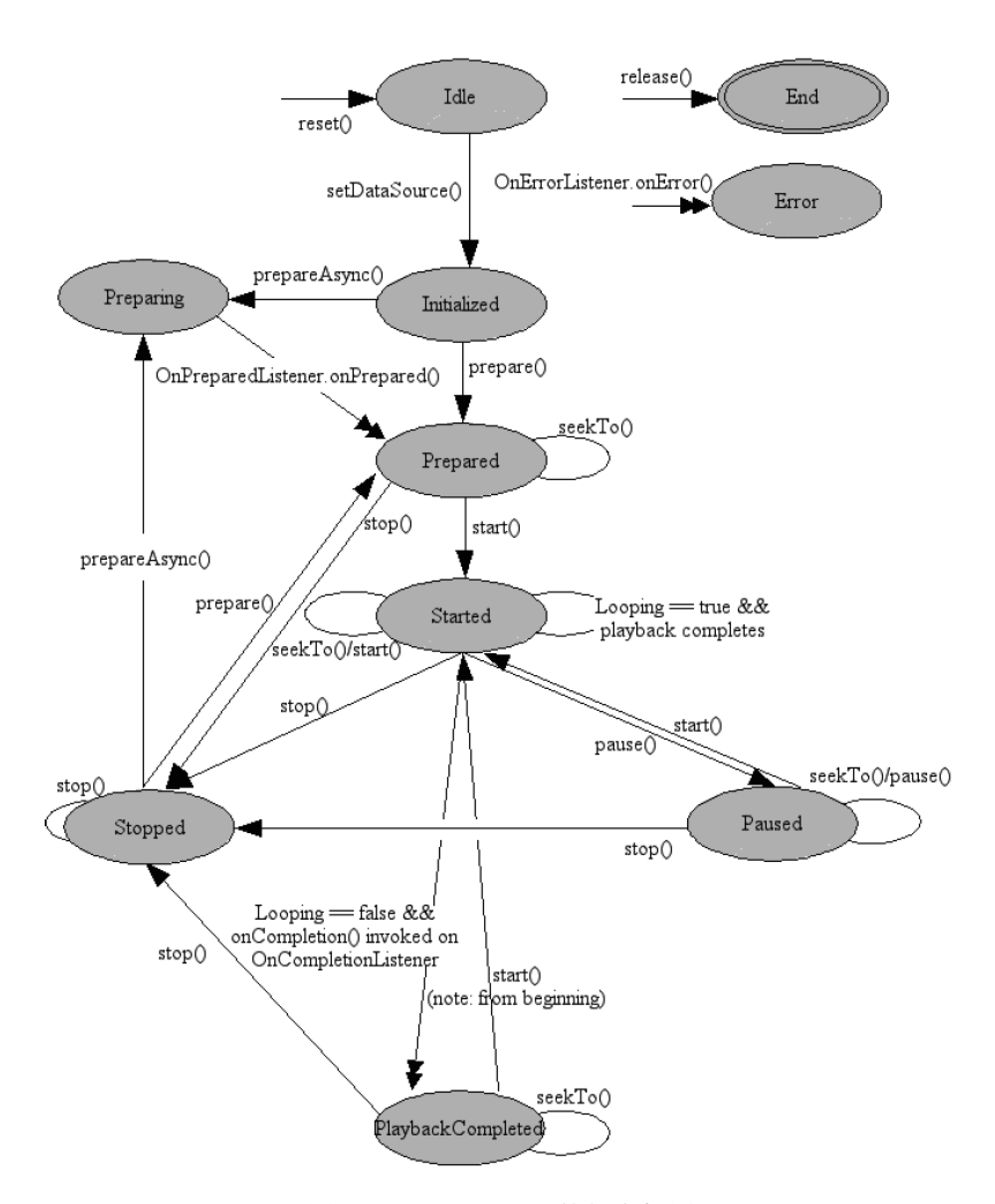


图 9-1　MediaPlayer 的状态机图

这里需要着重强调的是 MediaPlayer 是基于状态的。当写代码时，必须始终注意 MediaPlayer 所处的状态，因为 MediaPlayer 的方法都有其可以正常执行的有效状态。如果在一个错误的状态执行一个方法时，系统会抛出异常或产生不可预料的行为。

上面所述的 MediaPlayer 的状态机图明确指出哪些方法可以把 MediaPlayer 从一个状态迁移到另一个状态。例如，当新建一个 MediaPlayer 对象时（使用 new 操作），它处于 Idle 状态。在 Idle 状态时，通过调用 setDataSource()方法初始化 MediaPlayer 对象，迁移到 Initialized 状态。之后，使用 prepare()或 prepareAsync()方法准备 MediaPlayer 对象。当 MediaPlayer 对象准备就绪时，它进入 Prepared 状态，这时可以调用 start()方法来播放媒体文件。此时，MediaPlayer 对象处于 Started 状态，正如上图所示，可以通过调用 start()，pause()和 seekTo()方法，在 Started、Paused 和 PlaybackCompleted 三个状态之间进行迁移。当调用 stop()方法停止播放后，就不能再调用 start()方法播放媒体文件了，除非再次调用 prepare()方法准备 MediaPlayer 对象。

再强调一次，编写媒体播放代码时，一定注意 MediaPlayer 对象的状态，因为在错误

的状态调用方法会造成许多 bugs。

使用 MediaPlayer 播放时，注意不要在应用程序的 UI 主线程中准备 MediaPlayer。调用 prepare()方法通常会花费一定时间，因为它需要获取并解码媒体数据。因此，只要是任何需要花费一定时间执行的方法，都不要在应用程序的主 UI 线程中调用。那样做将会挂起 UI 主线程直到该方法执行完毕，这是一个非常差的用户体验，会造成应用程序不响应（Application Not Responding）错误。即使是很短的媒体数据，加载可能很快，但是记住任何花费超过 100 毫秒的操作都会造成 UI 界面发生明显地停顿并给用户特别明显的印象，应用程序响应很慢。

为了避免挂起 UI 主线程，创建一个新的线程准备 MediaPlayer，当准备就绪时，通知主 UI 线程。其实，用户不用自己编写新的线程逻辑，MediaPlayer 本身提供了一个非常方便的方法可以完成该任务，即 prepareAsync()方法，该方法在后台准备媒体数据，准备就绪后立刻返回。当媒体数据准备好后，MediaPlayer.OnPreparedListener 接口的 onPrepared()回调方法会自动被调用，以便继续后续处理。用户可以通过 MediaPlayer 的 setOnPreparedListener()方法注册监听器。

MediaPlayer 可能会耗尽系统资源。因此，应该采取措施避免 MediaPlayer 一直占用系统资源。当使用 MediaPlayer 播放完毕时，应该总是调用 release()方法，确保任何分配给它的系统资源被合适地释放。例如，如果应用的 Activity 收到 onStop()回调方法的调用，必须在其中释放 MediaPlayer，因为当该 Activity 不和用户进行交互时，MediaPlayer 还占用系统资源已经没有意义，除非正在后台播放。当 Activity 被继续或重启动时，需要创建一个新的 MediaPlayer，并再次准备它，之后才能继续播放。

```
mediaPlayer.release();
mediaPlayer = null;
```

一般来说，播放 MP3 音频文件需要遵循如下步骤。

（1）把将要播放的 MP3 文件放到 Android 项目的 res/raw 目录下，当然，也可以使用 URI 访问网络上的音频文件。

（2）通过调用 MediaPlayer.create()方法创建 MediaPlayer 的实例对象，并引用该 MP3 文件。

（3）调用 MediaPlayer 的 prepare()和 start()方法。

下面将通过一个例子演示如何播放一个 MP3 文件。首先，创建一个新的 Android 项目 MediaPlayerExample，创建一个 Activity，命名为 MediaPlayerActivity。接着，在项目的 res/文件夹下创建一个新的目录 raw，把将要播放的 MP3 文件放到该新建的文件夹下。然后，创建一个简单的按钮，用于控制 MP3 的播放，布局 XML 文件如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
>
<TextView
android:layout_width="fill_parent"
```

```
    android:layout height="wrap content"
    android:text="Simple Media Player"
    />
    <Button android:id="@+id/playsong"
    android:layout width="fill parent"
    android:layout height="wrap content"
    android:text="an MP3 audio file"
    />
    </LinearLayout>
```



接着，修改 MediaPlayerActivity.java 文件的代码，如下所示。

```
public class MediaPlayerActivity extends Activity {
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.main);
        Button mybutton = (Button) findViewById(R.id.playsong);
        mybutton.setOnClickListener(new Button.OnClickListener() {
           public void onClick(View v) {
               MediaPlayer mp =
        MediaPlayer.create(MediaPlayerActivity.this,
        R.raw.halotheme);
              mp.start();
              mp.setOnCompletionListener(new OnCompletionListener(){
              public void onCompletion(MediaPlayer arg0) {
                  }
              }
              );
           }
        }
    );
  }
}
```

如上所述，播放 MP3 文件相当简单。用户所做的所有事情就是使用布局 XML 文件创建的视图 View 显示界面，映射资源 ID “playsong” 到 mybutton，然后绑定到 setOnClickListener()方法。在该监听器方法中，创建使用 create(Context context, int resourceid)方法创建 MediaPlayer 的实例。最后，设置 setOnCompletionListener()方法，当播放完成后，执行某些资源释放的操作。当前例子中，我们没有做任何事情，不过，如果想要改变按钮状态，提供用户通知音乐播放完毕，或询问用户是否要播放另一首歌，你可以在该方法中做相应的处理。

9.1.3 播放视频

播放视频稍微复杂一些，因为还需要处理视频画面的显示。Android 平台提供一个 VideoView widget 来处理视频画面的显示，用户可以把它放到任何的界面布局管理器中。此外，Android 还提供大量的显示选项，包括缩放和着色。下面将通过一个例子演示如何

播放视频文件。首先，创建界面布局，如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout width="fill parent"
android:layout height="fill parent"
>
<VideoView android:id="@+id/video
android:layout width="320px"
android:layout height="240px"
/>/
</LinearLayout>
```

该布局界面中增加了一个 VideoView widget。它提供视频画面显示的功能，并且包含停止、播放、快进、快退等一些功能按钮。接着，编写一个类文件来播放视频，如下所示。

```
public class SimpleVideo extends Activity {
            private VideoView myVideo;
            private MediaController mc;
            public void onCreate(Bundle icicle) {
                super.onCreate(icicle);
                getWindow().setFormat(PixelFormat.TRANSLUCENT);
                setContentView(R.layout.main);
                myVideo = (VideoView) findViewById(R.id.video);
                File pathToTest= new File
           (Environment.getExternalFileDirectory(),"test.mp4");
                myVideo.setVideoPath(pathToTest);
                mc = new MediaController(this);
                mc.setMediaPlayer(myVideo);
                myVideo.setMediaController(mc);
                myVideo.requestFocus();
           }
}
```

在这个播放视频的类文件中，首先创建一个 translucent 窗口，这个窗口对于 SurfaceView 来说是必须的。接着，把 VideoView 作为播放视频的容器，并且使用 setVideoPath()方法设置将要播放的视频文件的路径。最后，创建 MediaController 类的实例对象，使用 setMediaController 方法向 VideoView 注册回调，以便视频播放完成后通知它。

运行该 Android 应用之前，需要使用 ADB 推送视频文件到 Android 设备上。当视频文件位于设备的 SD 卡上时，运行该应用并且触摸屏幕，控制按钮将会出现。

如上所示，VideoView 和 MediaPlayer 简化了视频的播放。不过需要注意的是，模拟器和真实设备在处理较大的视频文件时的反应有所不同。

9.1.4 录制音频

介绍完音视频播放后，接下来讲解音视频的录制。

录制音频的过程遵循以下 9 个步骤。

（1）创建 android.media.MediaRecorder 对象实例，以后所有的工作都是围绕该对象展开。

```
MediaRecorder mRecorder = new MediaRecorder();
```

（2）设置音频录制时采用的音频源（audio source）。这里，通过调用前面介绍的 android.media.MediaRecorder 对象的 MediaRecorder.setAudioSource()方法完成。MediaRecorder.AudioSource 是 MediaRecorder 的内部类，它主要定义智能手机常用的音频源资源，MediaRecorder.AudioSource.MIC 是最常用的音频源，代表手机的麦克风外设。除了 MediaRecorder.AudioSource.MIC 以外，还提供 VOICE_CALL、VOICE_DOWNLINK、VOICE_UPLINK 音频源，可以通过这些音频源进行语音通话的录制。

```
mRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
```

（3）设置输出音频数据的存储文件格式。这里，通过调用前面介绍的 MediaRecorder.setOutputFormat()方法完成。Android 平台支持的音频文件格式由 MediaRecorder.OutputFormat 内部类定义。以下为 Android 支持的主要文件格式。

- ❑ **MediaRecorder.OutputFormat.AMR_NB** 该常量表示输出文件是 AMR-NB 格式的语音文件，主要对人声进行编码。
- ❑ **MediaRecorder.OutputFormat.MPEG_4** 该常量表示输出文件是 MPEG-4 格式的多媒体文件，其中，可能同时包含音频和视频信息。
- ❑ **MediaRecorder.OutputFormat.THREE_GPP** 该常量表示输出文件是 3GPP 格式的文件，其中，可能同时包含音频和视频信息。

```
mRecorder.setOutputFormat(MediaRecorder.OutputFormat.AMR_NB);
```

（4）指定输出音频数据存放的文件。这里，通过调用前面介绍的 MediaRecorder.setOutputFile()方法完成，该方法可以接受两种参数：文件描述符（FileDescriptor）和文件路径字符串。

```
FileDescriptor fd = ...;
mRecorder.setOutputFile(fd);
```

或者

```
String mFileName = ...;
mRecorder.setOutputFile(mFileName);
```

（5）设置音频编码器。这里，通过调用前面介绍的 MediaRecorder. setAudioEncoder()方法完成。Android 平台支持的音频编码器由 MediaRecorder.AudioEncoder 内部类定义。最常用的音频编码器是 MediaRecorder.AudioEncoder.AMR_NB。AMR_NB 是自适应多速率窄带音频编解码器，该编解码器主要针对语音数据进行优化，使其不适应语音之外的其他音频信息。

```
mRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
```

（6）调用 MediaRecorder.prepare()方法，准备工作就绪，可以开始录制音频。

```
mRecorder.prepare();
```



（7）调用 MediaRecorder.start()方法，开始录制。

```
mRecorder.start();
```

（8）录制完成之后，要停止录制，需要调用 MediaRecorder.stop()方法。

```
mRecorder.stop();
```

（9）还需要调用 MediaRecorder.release()方法，释放所占用的资源。到这里，整个录制过程就完成了。

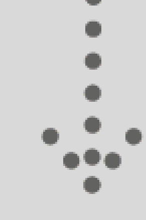
下面将通过一个例子演示如何录制音频文件。首先，创建一个 Android 项目 SoundRecordingExample。接着，编辑 AndroidManifest.xml 文件，添加如下内容。

```
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_
STORAGE"/>
```

上面的代码设置应用程序的权限，允许应用录制音频文件并播放它们。接下来，创建如下所示的类。

```
public class SoundRecordingDemo extends Activity {
    MediaRecorder mRecorder;
    File mSampleFile = null;
    static final String SAMPLE_PREFIX = "recording";
    static final String SAMPLE_EXTENSION = ".3gpp";
    private static final String OUTPUT_FILE = "/sdcard/audiooutput.3gpp";
    private static final String TAG = "SoundRecordingExample";

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        this.mRecorder = new MediaRecorder();
        Button startRecording = (Button) findViewById(R.id.
        startrecording);
        Button stopRecording = (Button) findViewById(R.id.stoprecording);
        startRecording.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
        startRecording();
        }
    });
    stopRecording.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            stopRecording();
            addToDB();
        }
    });
}
protected void addToDB() {
    ContentValues values = new ContentValues(3);
    long current = System.currentTimeMillis();
    values.put(MediaColumns.TITLE, "test audio");
    values.put(MediaColumns.DATE_ADDED, (int) (current / 1000));
    values.put(MediaColumns.MIME_TYPE, "audio/3gpp");
    values.put(MediaColumns.DATA, OUTPUT_FILE);
```



```
        ContentResolver contentResolver = getContentResolver();
        Uri base = MediaStore.Audio.Media.EXTERNAL CONTENT URI;
        Uri newUri = contentResolver.insert(base, values);
        sendBroadcast(new Intent(Intent.ACTION MEDIA SCANNER SCAN FILE,
        newUri)); }

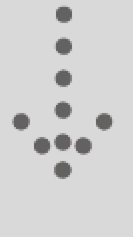
    protected void startRecording() {
        this.mRecorder = new MediaRecorder();
        this.mRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
        this.mRecorder.setOutputFormat(MediaRecorder.
    OutputFormat.THREE GPP);
        this.mRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR NB);
        this.mRecorder.setOutputFile(OUTPUT FILE);
        try {
            this.mRecorder.prepare();
        } catch (IllegalStateException e1) {
            // TODO Auto-generated catch block
        e1.printStackTrace();
        } catch (IOException e1) {
            // TODO Auto-generated catch block
        e1.printStackTrace();
    }
    this.mRecorder.start();
    if (this.mSampleFile == null) {
        File sampleDir = Environment.getExternalStorageDirectory();
        try {
            this.mSampleFile =
    File.createTempFile(SoundRecordingDemo.SAMPLE PREFIX,
    SoundRecordingDemo.SAMPLE EXTENSION, sampleDir);
        } catch (IOException e) {
            Log.e(SoundRecordingDemo.TAG, "sdcard access error");
            return;
        }
      }
    }

    protected void stopRecording() {
        this.mRecorder.stop();
        this.mRecorder.release();
      }
    }
```

代码的第一部分创建按钮和按钮监听器，监听开始和停止录制的事件，引用 main.xml 文件。代码的第一部分有一个重要的方法 addToDB()，该方法用于设置音频文件的元数据信息，包括标题、日期、文件类型。接下来，调用 Intent ACTION_MEDIA_SCANNER_SCAN_FILE 通知应用程序，一个新的音频文件已经创建。使用该 Intent 允许用户查询播放列表中的新的音频文件并播放它们。

接下来创建 startRecording()方法，该方法首先创建一个新的 MediaRecorder 实例，然后设置音频源和麦克风，设置输出格式为 THREE_GPP，设置音频编码器为 AMR_NB，然后设置输出文件路径。接着，调用 prepare()和 start()方法开始录制音频。最后，创建 stopRecording()方法，调用 stop()和 release()方法停止 MediaRecorder 录制音频。

所有工作准备就绪，构建该应用程序，并且在模拟器中运行它，单击开始录制按钮，几秒过后，单击停止录制按钮，打开 DDMS，可以在 sdcard 目录下看到录制的文件。可以使用设备的媒体播放器，文件浏览器查看并播放该录制的文件。



9.1.5 录制视频

不同于录制音频，视频录制时，Android 要求在录制之前必须先预览视频，这可能对某些应用有些不合适，但这是 Android 2.2 及其以上版本的必须要求。

正如录制音频一样，录制视频之前，必须设置一些权限，包括 RECORD_VIDEO、CAMERA、RECORD_AUDIO 和 WRITE_EXTERNAL_STORAGE。下面将通过一个例子演示如何录制视频文件。首先，创建一个 Android 项目 VideoCamExample，接着，编辑 AndroidManifest.xml 文件，添加如下内容。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.msi.manning.chapter10.VideoCam"
android:versionCode="1"
android:versionName="1.0">
<application android:icon="@drawable/icon"
android:label="@string/app name">
<activity android:name=".VideoCam"
android:label="@string/app name">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name=
"android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
</application>
<uses-permission android:name="android.permission.CAMERA">
</uses-permission>
<uses-permission android:name=
"android.permission.RECORD_AUDIO"></uses-permission>
<uses-permission android:name=
"android.permission.RECORD VIDEO"></uses-permission>
<uses-permission android:name=
"android.permission.WRITE EXTERNAL STORAGE" />
<uses-feature android:name="android.hardware.camera" />
</manifest>
```

该文件最后使用了 uses-feature 语句，目的是指明该程序的运行依赖于哪些软硬件资源，本例中指明需要使用摄像头。

接下来，为该应用创建一个简单的布局，包含预览区域、开始、停止、暂停、播放按钮。布局 XML 文件如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/
```

```
android"
android:orientation="vertical"
android:layout width="fill parent"
android:layout height="fill parent">
<RelativeLayout android:layout width="fill parent"
android:layout height="wrap content"
android:id="@+id/relativeVideoLayoutView"
android:layout centerInParent="true">
<VideoView android:id="@+id/videoView"
android:layout width="176px"
android:layout height="144px"
android:layout centerInParent="true"/>
</RelativeLayout>
<LinearLayout
android:layout width="wrap content"
android:layout height="wrap content"
android:orientation="horizontal"
android:layout centerHorizontal="true"
android:layout below="@+id/relativeVideoLayoutView">
<ImageButton android:id="@+id/playRecordingBtn"
android:layout width="wrap content"
android:layout height="wrap content"
android:background="@drawable/play"
/>
<ImageButton android:id="@+id/bgnBtn"
android:layout width="wrap content"
android:layout height="wrap content"
android:background="@drawable/record"
android:enabled="false"
/>
</LinearLayout>
</RelativeLayout>
```

视频录制采用与音频录制相似的步骤，同时加上与视频相关的特殊步骤。

首先需要创建 MediaRecorder 对象，然后依次进行后续的操作。

```
MediaRecorder video_recorder = new MediaRecorder();
```

1）设置音频和视频源

创建 MediaRecorder 对象后，需要设置音频和视频源。可以使用 setAudioSource()方法设置音频源，传入一个想要使用的音频源常量，设置方法已在 12.1 节音频录制中介绍过，此处不再重复叙述。为了设置视频源，可以使用 setVideoSource()方法。可能的视频源的值定义在 MediaRecorder.VideoSource 类的常量，其中只包含两个常量：CAMERA 和 DEFAULT。其实这两个常量表示含义相同，都是指设备上的主摄像头。

```
video_recorder.setVideoSource(MediaRecorder.VideoSource.DEFAULT);
```

2）输出格式

设置音频和视频源之后，可以使用 MediaRecorder 的 setOutputFormat()方法设置输出格式，传入要使用的格式。

```
video_recorder.setOutputFormat(MediaRecorder.OutputFormat.DEFAULT);
```

可能的格式定义在 MediaRecorder.OutputFormat 中的常量。

- **DEFAULT** 使用默认的输出格式，默认的输出格式根据设备的不同而不同。
- **MPEG_4** 指定音频和视频被录制在一个 MPEG-4 格式的文件中，扩展名是.mp4。MPEG-4 文件通常包含 H.264、H.263 或 MPEG-4 Part 2 编码的视频，以及 AAC 或 MP3 编码的音频。MPEG-4 文件被广泛用于许多其他在线视频技术或消费电子设备上。
- **THREE_GPP** 指定音频和视频将被录制到一个 3GPP 格式的文件中，扩展名是.3gp。3GPP 文件通常包含使用 H.264、H.263 或 MPEG-4 Part 2 编码的视频和使用 AMR 或 AAC 编码的音频。

3）设置音频和视频编解码器

设置输出格式之后，需要指定想要使用的音频和视频编解码器。可以使用 MediaRecorder 的 setVideoEncoder()方法设置视频编解码器。

```
video.setVideoEncoder(MediaRecorder.VideoEncoder.DEFAULT);
```

可以使用的编解码器定义在 MediaRecorder.VideoEncoder 中的常量：

- **DEFAULT** 使用默认的视频编解码器。多数情况下是 H.263，Android 设备上唯一必须支持的编解码器。
- **H263** 指定 H.263 为视频编解码器。H.263 是在 1995 年发布的编解码器，专门为低比特率视频传输而开发。它是许多早期 Internet 视频技术的基础，如 Flash 和 RealPlayer 早期使用的技术。在 Android 平台它是必须支持的编码方式，因此可以放心地使用。
- **H264** 指定 H.264 为视频编解码器。H.264 是当前最先进的编解码器，广泛应用于各种技术，从 BlueRay 到 Flash。
- **MPEG_4_SP** 指定视频编解码器为 MPEG-4 SP。MPEG-4 SP 是 MPEG-4 Part 2 Simple Profile，它发布于 1999 年，为需要低比特率视频且不需要大处理器能力的技术而开发。

对于音频部分，可以使用 setAudioEncoder()方法设置音频编解码器，设置方法在 12.1 节音频录制中介绍过，此处不再重复叙述。

4）设置音频和视频比特率

使用 MediaRecorder 的 setVideoEncodingBitRate()方法设置视频编码比特率。视频的低比特率设置在 256000 位/秒（256kbps）范围之内，而高比特率在 3000000 位/秒（3mbps）范围之内。

```
video_recorder.setVideoEncodingBitRate(150000);
```

使用 MediaRecorder 的 setAudioEncodingBitRate()方法设置音频编码比特率。8000 位/秒是一个非常低的比特率，适合于在慢速网络上实时传输的音频，而 196000 位/秒在 MP3 文件中很常见。

```
video_recorder.setAudioEncodingBitRate (8000);
```

5）设置音频采样率

和编码比特率相同，音频采样率对于音频的质量也非常重要。可以使用 MediaRecorder

的 setAudioSamplingRate()方法设置音频采样率。采样率以 Hz 为单位，表示每秒采样的数量。采样率越高，则在录制音频文件中可以表示的音频频率的范围越大。一个低端的采样率 8000Hz 适合于录制低质量的音频，而高端的采样率 48000Hz 可用于 DVD 和许多其他高质量的视频格式。

```
video_recorder. setAudioSamplingRate (8000);
```

6）设置音频通道

可以使用 setAudioChannels()方法指定将要录制的音频通道的数量。目前，音频大都限制为大多数 Android 设备上的单一通道麦克风，因此使用一个以上的通道不会有益处。对于通道数量，一般是单声道为一个通道，而立体声为两个通道。

```
video_recorder. setAudioChannels (1);
```

7）设置视频帧速率

可以使用 setVideoFrameRate()方法来控制每秒录制的视频帧数。每秒 12～15 帧之间的值通常足以表示运动。具体使用的实际帧率取决于设备的能力。

```
video_recorder. setVideoFrameRate(15);
```

8）设置视频大小

可以通过 setVideoSize()方法并传入宽高值来控制录制的视频的宽度和高度。标准大小的范围是 176*144～640*480，许多设备甚至支持更高的分辨率。

```
video_recorder. setVideoSize (640, 480);
```

9）设置最大文件大小

使用 setMaxFileSize()方法设置最大文件大小，单位为字节。

```
video_recorder. setMaxFileSize (10000000); //10MB
```

为了确定是否已达到最大文件大小。需要在 Activity 中实现 MediaRecorder.OnInfoListener，同时在 MediaRecorder 中注册它。然后系统会调用 onInfo 方法，检查 what 参数是否等于 MediaRecorder.MEDIA_RECORDER_INFO_FILESIZE_REACHED。

10）设置持续时间

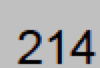
使用 setMaxDuration()方法设置最长持续时间，单位为毫秒。

```
video_recorder. setMaxDuration(10000); //10 秒
```

为了确定是否已达到最长持续时间。需要在 Activity 中实现 MediaRecorder.OnInfoListener，同时在 MediaRecorder 中注册它。当已达到最长持续时间时就会触发 onInfo() 方法，检查 what 参数是否等于 MediaRecorder.MEDIA_RECORDER_INFO_MAX_DURATION_REACHED。

11）概要

MediaRecorder 有一个 setProfile()方法，接受 CamcorderProfile 实例作为参数。使用该



方法允许根据预设值设置整个配置变量集。其中，CamcorderProfile.QUALITY_LOW 指低质量视频捕获设置，CamcorderProfile.QUALITY_HIGH 指高质量视频捕获设置。

12）输出文件

使用 setOutputFile()方法设置输出文件的位置。

```
video_recorder. setOutputFile ("/sdcard/video_recorded.mp4"); //10 秒
```

13）预览表面

由于是视频录制，录制过程中需要看到画面。因此，需要为 MediaRecorder 指定一个取景器以预览要绘制的图像。需要使用 SurfaceView 和 SurfaceHolder.Callback，将在下面的例子中进行介绍。

14）准备录制

设置好 MediaRecorder 实例后，就可以使用 prepare()方法准备 MediaRecorder 了。

```
video_recorder.prepare();
```

15）开始录制

MediaRecorder 实例准备好后，就可以开始录制了。

```
video_recorder.start();
```

16）停止录制

录制过程中，可以通过 stop()方法停止录制。

```
video_recorder.stop();
```

17）释放资源

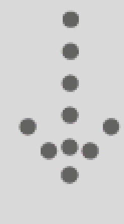
最后，不要忘记需要释放占用的资源。

```
video_recorder.release();
```

如上所述，视频录制类似于音频录制。接下来，创建 VideoCamActivity 类来完成项目，代码如下所示。

```
public class VideoCam extends Activity implements SurfaceHolder.Callback
{
        private MediaRecorder recorder = null;
        private static final String OUTPUT FILE =
        "/sdcard/uatestvideo.mp4";
        private static final String TAG = "RecordVideo";
        private VideoView videoView = null;
        private ImageButton startBtn = null;
        private ImageButton playRecordingBtn = null;
        private Boolean playing = false;
        private Boolean recording = false;

        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
            startBtn = (ImageButton) findViewById(R.id.bgnBtn);
            playRecordingBtn = (ImageButton)
```



```
        findViewById(R.id.playRecordingBtn);
        videoView = (VideoView)this.findViewById(R.id.videoView);
        final SurfaceHolder holder = videoView.getHolder();
        holder.addCallback(this);
        holder.setType(SurfaceHolder.SURFACE TYPE PUSH BUFFERS);
        startBtn.setOnClickListener(new OnClickListener() {
        public void onClick(View view) {
            if(!VideoCam.this.recording & !VideoCam.this.playing)
            {
                try
                {
                    beginRecording(holder);
                    playing=false;
                    recording=true;
                    startBtn.setBackgroundResource(R.drawable.stop);
                    } catch (Exception e) {
                        Log.e(TAG, e.toString());
                        e.printStackTrace();
                    }
              }
              else if(VideoCam.this.recording)
              {
                  try
                  {
                      stopRecording();
                      playing = false;
                      recording= false;
                      startBtn.setBackgroundResource(R.drawable.play);
                  }catch (Exception e) {
                      Log.e(TAG, e.toString());
                      e.printStackTrace();
                  }
              }
        }
    });

    playRecordingBtn.setOnClickListener(new OnClickListener() {
        public void onClick(View view)
        {
            if(!VideoCam.this.playing & !VideoCam.this.recording)
            {
               try
               {
                    playRecording();
                    VideoCam.this.playing=true;
                    VideoCam.this.recording=false;
                    playRecordingBtn.setBackgroundResource
    (R.drawable.stop);
               } catch (Exception e) {
                   Log.e(TAG, e.toString());
                   e.printStackTrace();
               }
            }
            else if(VideoCam.this.playing)
            {
                  try
                  {
```

```
                            stopPlayingRecording();
                            VideoCam.this.playing = false;
                            VideoCam.this.recording= false;
                            playRecordingBtn.setBackgroundResource
              (R.drawable.play);
                      }catch (Exception e) {
                          Log.e(TAG, e.toString());
                          e.printStackTrace();
                      }
                 }
            }
    });
    }

    public void surfaceCreated(SurfaceHolder holder) {
           startBtn.setEnabled(true);
    }

    public void surfaceDestroyed(SurfaceHolder holder) {
    }

    public void surfaceChanged(SurfaceHolder holder, int format, int width, int
    height) {
        Log.v(TAG, "Width x Height = " + width + "x" + height);
    }

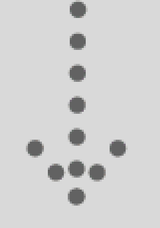
    private void playRecording() {
       MediaController mc = new MediaController(this);
       videoView.setMediaController(mc);
       videoView.setVideoPath(OUTPUT FILE);
       videoView.start();
    }

    private void stopPlayingRecording() {
       videoView.stopPlayback();
    }

    private void stopRecording() throws Exception {
       if (recorder != null) {
          recorder.stop();
       }
    }

    protected void onDestroy() {
       super.onDestroy();
       if (recorder != null) {
          recorder.release();
       }
    }

    private void beginRecording(SurfaceHolder holder) throws Exception
    {
       if(recorder!=null)
       {
         recorder.stop();
         recorder.release();
```



```
        }
        File outFile = new File(OUTPUT_FILE);
        if(outFile.exists())
        {
            outFile.delete();
        }
        try {
             recorder = new MediaRecorder();
             recorder.setVideoSource(MediaRecorder.VideoSource.CAMERA);
             recorder.setAudioSource(MediaRecorder.AudioSource.MIC);
             recorder.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);
             recorder.setVideoSize(320, 240);
             recorder.setVideoFrameRate(15);
             recorder.setVideoEncoder(MediaRecorder.VideoEncoder.MPEG_4_SP);
             recorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
             recorder.setMaxDuration(20000);
             recorder.setPreviewDisplay(holder.getSurface());
             recorder.setOutputFile(OUTPUT_FILE);
             recorder.prepare();
             recorder.start();
         }
         catch(Exception e) {
             Log.e(TAG, e.toString());
             e.printStackTrace();
         }
       }
    }
```

除了开头的设置成员之外，要做的第一件事是创建显示 surface 用于支持摄像头预览。然后创建重要的 beginRecording()方法，该方法确保所有准备工作都已就绪，可以开始录制视频。首先，确保摄像头可用，然后，检测输出文件是否存在，存在则删除，接着，遵循录制视频的一般流程，建立并设置 MediaRecorder。

9.2　使用电话 API

基于 Android 平台的智能设备，特别是智能手机，都支持在程序中使用电话相关的服务。本节介绍如何使用电话 API 编写拨打电话、发送 SMS、接收 SMS 的功能。

9.2.1　拨打电话

拨打电话最简单的方法是使用 Intent.ACTION_CALL 动作调用 Android 内部的拨号应用，该方法调用拨号应用，把电话号码传给拨号应用，显示在拨号盘上，开始拨打。另外，还可以通过使用 Intent.ACTION_DIAL 动作来调用 Android 内部的拨号应用，该方式也是首先调用拨号应用，填充电话号码，显示在拨号盘上，但是不发起拨打电话的动作，而是等待用户单击拨号盘上的拨打按钮。下面代码演示了如何使用 Intent 动作拨打电话。

```
dialintent = (Button) findViewById(R.id.dialintent button);
dialintent.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        Intent intent = new Intent(Intent.DIAL ACTION,
        Uri.parse("tel:" + NUMBER));
        startActivity(intent);
    }
});

callintent = (Button) findViewById(R.id.callintent button);
callintent.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        Intent intent =
        new Intent(Intent.CALL ACTION, Uri.parse("tel:" + NUMBER));
        startActivity(intent);
    }
});
```

注意，使用 Intent.ACTION_DIAL 拨打电话不需要特殊的权限，但使用 Intent.ACTION_CALL 拨打电话需要给应用程序添加 CALL_PHONE 的权限。

9.2.2 发送 SMS

Android SDK 提供给开发人员发送和接收 SMS 的一系列 API。本小节将通过例子讲解如何编写程序发送 SMS。为了发送 SMS，首先需要向应用程序的 manifest 文件中添加 android.permission.SEND_SMS 权限，然后使用 android.telephony.SmsManager 类。

首先，创建一个 Android 项目 SMSExample，接着，编辑布局 XML 文件，添加如下内容。

```
<?xml version="1.0" encoding="utf-8"?>
<!-- This file is /res/layout/main.xml -->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout width="fill parent"

android:layout height="fill parent">
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="horizontal"
android:layout width="fill parent"
android:layout height="wrap content">
<TextView android:layout width="wrap content"
android:layout height="wrap content"
android:text="Destination Address:" />
<EditText android:id="@+id/addrEditText"
android:layout width="fill parent"
android:layout height="wrap content"
android:phoneNumber="true"
```

```
android:text="9045551212" />
</LinearLayout>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout width="fill parent"
android:layout height="wrap content">
<TextView android:layout width="wrap content"
android:layout height="wrap content"
android:text="Text Message:" />
<EditText android:id="@+id/msgEditText"
android:layout width="fill parent"
android:layout height="wrap content"
android:text="hello sms" />
</LinearLayout>
<Button android:id="@+id/sendSmsBtn"
android:layout_width="wrap_content"
android:layout height="wrap content"
android:text="Send Text Message"
android:onClick="doSend" />
</LinearLayout>
```

然后，创建 SendSMSActivity.java 文件，填写如下内容。

```
import android.app.Activity;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;
public class SendSMSActivity extends Activity
{

   @Override
   protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.main);
   }

   public void doSend(View view) {
       EditText addrTxt =
       (EditText) findViewById(R.id.addrEditText);
       EditText msgTxt =
       (EditText) findViewById(R.id.msgEditText);
   try {
       sendSmsMessage(
           addrTxt.getText().toString(),
           msgTxt.getText().toString());
       Toast.makeText(this, "SMS Sent",
         Toast.LENGTH LONG).show();
   } catch (Exception e) {
```

```
            Toast.makeText(this, "Failed to send SMS",
                Toast.LENGTH LONG).show();
            e.printStackTrace();
        }
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
    }

    private void sendSmsMessage(String address,String message)throws Exception
    {
        SmsManager smsMgr = SmsManager.getDefault();
        smsMgr.sendTextMessage(address, null, message, null, null);
      }
    }
```

界面布局 XML 文件中添加了两个 EditText 文本编辑框，一个用于获取 SMS 的接收者的地址（电话号码），另一个用于获取要发送的短消息。Android SDK 提供给开发人员发送和接收 SMS 的一系列 API。此外，用户界面上还有一个按钮，用于触发发送 SMS 的动作。

代码中最重要的一个方法是 sendSmsMessage()，该方法内部使用 SmsManager 类的 sendTextMessage()方法发送 SMS 短消息。

我们可以在模拟器中测试 SMS 发送。首先启动该程序的两个实例，使用其中一个模拟器的端口号作为目的地址，该端口号就是出现在模拟器窗口标题栏的号码，它通常是类似 5554 这样的号码。单击发送按钮后，会看见一个通知消息出现在另一个模拟器上，表明 SMS 已经发送成功。

9.2.3 接收 SMS

介绍完 SMS 发送之后，本小节接着通过扩展上节的例子讲解如何编写程序接收 SMS。通过添加一个新的 BroadcastReceiver 派生类来监听 android.provider.Telephony.SMS_RECEIVED 这个动作。这个动作是在当一个 SMS 短消息被设备接收后由 Android 平台自动广播出去的。当用户在该动作上注册了自己的 receiver 后，每当 SMS 短消息收到后，应用就会被通知。首先，仍然需要向应用程序的 manifest 文件中添加 android.permission.RECEIVE_SMS 权限，如下所示。

```
<?xml version="1.0" encoding="utf-8"?>
<!-- This file is AndroidManifest.xml -->
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.androidbook.telephony" android:versionCode="1"
android:versionName="1.0">
<application android:icon="@drawable/icon"
android:label="@string/app name">
<activity android:name=".SMSExample"
android:label="@string/app_name">
```

```
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<receiver android:name="SMSMonitorEx">
<intent-filter>
<action
android:name="android.provider.Telephony.SMS_RECEIVED"/>
</intent-filter>
</receiver>
</application>
<uses-sdk android:minSdkVersion="4" />
<uses-permission android:name="android.permission.SEND_SMS"/>
<uses-permission android:name="android.permission.RECEIVE_SMS"/>
</manifest>
```

创建一个 Android 项目 SMSExample，接着，编辑布局 XML 文件，添加如下内容。

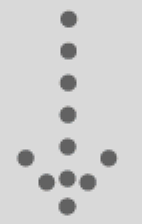
```
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.telephony.SmsMessage;
import android.util.Log;
public class SMSMonitorEx extends BroadcastReceiver
{
    private static final String ACTION =
    "android.provider.Telephony.SMS_RECEIVED";

    @Override
    public void onReceive(Context context, Intent intent)
    {
       if(intent!=null && intent.getAction()!=null &&
    ACTION.compareToIgnoreCase(intent.getAction())==0)
       {
          Object[] pduArray= (Object[])
             intent.getExtras().get("pdus");
          SmsMessage[] messages = new
             SmsMessage[pduArray.length];
          for (int i = 0; i<pduArray.length; i++) {
             messages[i] = SmsMessage.createFromPdu(
                (byte[])pduArray [i]);
             Log.d("SMSMonitorEx", "From: " +
                messages[i].getOriginatingAddress());
             Log.d("MySMSMonitor", "Msg: " +
                messages[i].getMessageBody());
          }
          Log.d("MySMSMonitor","SMS Message Received.");
       }
```

```
        }
    }
```

9.3 本章小结

本章讲解了如何在 Android 平台下进行音视频录制和播放及拨打电话、收发短消息的相关知识和方法。同时，通过一个具体的实例，可以使读者能够真正掌握相关功能的开发技术。



第 10 章　网络与通信

目前，用户可以使用手机查看电子邮件、浏览网页、观看网络视频、在线听音乐，这些都归功于智能手机提供的强大的网络通信功能。本章主要介绍 Android 平台的网络开发的基础知识，并讲解如何通过 HTTP 和 Socket 等进行网络应用的开发。

10.1　网络概述

绝大多数情况下，编写 Android 程序的 API 抽象了底层的网络细节，这使编程变得很方便，这样用户就可以专注于应用程序，而不是担心关于路由、可靠的包交付等等。然而，理解网络的底层运行机制有助于更好地设计和调试应用程序。

（1）节点

网络中的设备必须有确切的地址，才能相互之间发送数据。每个有地址的设备叫做一个网络节点。节点可以是普通的计算机，服务器，智能手机，智能手表等等，只要它们有网络协议栈就可以提供网络相关功能。

（2）协议层

协议是预先定义好的双方进行通信的一系列规则。TCP/IP 协议栈分为 4 层：

链路层——物理设备地址解析协议，例如 ARP 和 RARP。

网络层——包括多个协议，ping 协议，ICMP 协议，IP 协议等。

传输层——各种类型的转发协议，例如 TCP 和 UDP。

应用层——HTTP、FTP、SMTP、DNS 等。

协议层是对网络协议栈的不同等级的抽象。最底层链路层直接和物理设备相关，负责网络数据通过物理线路在设备间进行传播；网络层；特别是 IP 协议，主要负责寻址和路由的功能；传输层负责数据报文的转发细节；最后，最底层的应用层协议，负责发送接收与具体应用相关的数据。

10.2　HTTP 网络开发

使用手机访问互联网的最常见方式是通过 HTTP 协议。它是一种网络应用层协议。使用 HTTP 协议可以执行各种任务，例如，从 Web 服务器上下载 Web 页面，下载二进制数据等等。下面将创建一个项目，演示如何使用 HTTP 协议连接到 Web 服务器，进而下载各种内容。

首先，使用 Eclipse 创建一个新的 Android 项目，命名为 net_demo。

然后，修改 AndroidManifest.xml 文件，代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="net.learn2develop.Networking"
android:versionCode="1"
android:versionName="1.0" >
<uses-sdk android:minSdkVersion="14" />
<uses-permission android:name="android.permission.INTERNET"/>
<application
android:icon="@drawable/ic launcher"
android:label="@string/app name" >
<activity
android:label="@string/app name"
android:name=".NetworkingActivity" >
<intent-filter >
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
</application>
</manifest>
```

最后，修改 NetworkingActivity.java 文件，代码如下：

```
import android.app.Activity;
import android.os.Bundle;
import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLConnection;
import android.util.Log;

public class NetworkingActivity extends Activity {

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
}

private    InputStream    OpenHttpConnection(String    urlString)    throws
IOException
{
InputStream in = null;
int response = -1;
URL url = new URL(urlString);
URLConnection conn = url.openConnection();
if (!(conn instanceof HttpURLConnection))
```

```
throw new IOException("Not an HTTP connection");
try{
HttpURLConnection httpConn = (HttpURLConnection) conn;
httpConn.setAllowUserInteraction(false);
httpConn.setInstanceFollowRedirects(true);
httpConn.setRequestMethod("GET");
httpConn.connect();
response = httpConn.getResponseCode();
if (response == HttpURLConnection.HTTP_OK) {
in = httpConn.getInputStream();
}
}
catch (Exception ex)
{
Log.d("Networking", ex.getLocalizedMessage());
throw new IOException("Error connecting");
}
return in;
}
}
```

使用 HTTP 协议连接网络时，应用程序需要 Internet 权限，因此，首先要在 AndroidManifest.xml 文件中添加 Internet 权限。

接下来定义了 OpenHttpConnection 方法，接收 URL 字符串作为参数，然后返回 InputStream 对象。使用 InputStream 对象可以下载各种类型的数据。在该方法中，使用 HttpURLConnection 对象打开 HTTP 连接，然后设置该连接的各种属性，代码如下所示：

```
HttpURLConnection httpConn = (HttpURLConnection) conn;
httpConn.setAllowUserInteraction(false);
httpConn.setInstanceFollowRedirects(true);
httpConn.setRequestMethod("GET");
```

尝试与 Web 服务器建立连接之后，HTTP 响应代码会返回。如果连接建立成功，返回的响应码是 HTTP_OK，然后，从该连接中可以得到一个 InputStream 对象：

```
httpConn.connect();
response = httpConn.getResponseCode();
if (response == HttpURLConnection.HTTP_OK) {
in = httpConn.getInputStream();
}
```

使用 InputStream 对象可以从 Web 服务器上下载数据。

一个常见的任务是从 Web 服务器上下载二进制数据。例如，从服务器上下载一副图片并显示。下面的代码演示如何完成这一任务。

首先，使用前面创建的项目，用下面的代码替代原有的 main.xml 文件。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
```

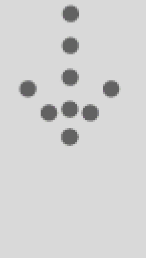
```
android:layout height="fill parent"
android:orientation="vertical" >

<ImageView
android:id="@+id/img"
android:layout width="wrap content"
android:layout height="wrap content"
android:layout gravity="center" />
</LinearLayout>
```

然后，用下面的代码替换 NetworkingActivity.java 文件。

```
import android.widget.ImageView;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.AsyncTask;
public class NetworkingActivity extends Activity {
ImageView img;
private   InputStream   OpenHttpConnection(String   urlString)   throws
IOException
{
InputStream in = null;
int response = -1;
URL url = new URL(urlString);
URLConnection conn = url.openConnection();
if (!(conn instanceof HttpURLConnection))
throw new IOException("Not an HTTP connection");
try{
HttpURLConnection httpConn = (HttpURLConnection) conn;
httpConn.setAllowUserInteraction(false);
httpConn.setInstanceFollowRedirects(true);
httpConn.setRequestMethod("GET");
httpConn.connect();
response = httpConn.getResponseCode();
if (response == HttpURLConnection.HTTP OK) {
in = httpConn.getInputStream();
}
}
catch (Exception ex)
{
Log.d("Networking", ex.getLocalizedMessage());
throw new IOException("Error connecting");
}
return in;
}

private Bitmap DownloadImage(String URL)
{
Bitmap bitmap = null;
InputStream in = null;
try {
```



```
in = OpenHttpConnection(URL);
bitmap = BitmapFactory.decodeStream(in);
in.close();
} catch (IOException e1) {
Log.d("NetworkingActivity", e1.getLocalizedMessage());
}
return bitmap;
}

private class DownloadImageTask extends AsyncTask<String, Void, Bitmap>
{

protected Bitmap doInBackground(String... urls) {
return DownloadImage(urls[0]);
}
protected void onPostExecute(Bitmap result) {
ImageView img = (ImageView) findViewById(R.id.img);
img.setImageBitmap(result);
}
}

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
new DownloadImageTask().execute(
"http://www.mayoff.com/5-01cablecarDCP01934.jpg");
}
}
```

最后，按 F11 键在模拟器上调试应用。图 10-1 显示了从网络上下载的这幅图片，用 ImageView 显示出来。

在 Android 3.0 之后，所有同步代码都不允许放到主 UI 线程中执行，而是必须用 AsyncTask 类包装起来。使用 AsyncTask 类可以在单独的 UI 线程中执行后台任务，而不影响主 UI 线程的执行。所有需要异步执行的代码放到 doInBackground()方法中，而当任务完成时，结果通过 onPostExecute()方法传回。

除了下载二进制数据外，还可以下载普通文本数据，例如普通网页数据。具体实现细节与下载二进制数据类似，下面的代码演示了如何从网站上下载一个网页。

首先，仍然使用前面创建的项目，添加如下代码到 NetworkingActivity.java 文件中。



图 10-1　通过 HTTP 下载图片

```
import java.io.InputStreamReader;
```

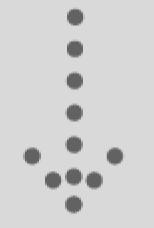
```
private String DownloadText(String URL)
{
int BUFFER SIZE = 2000;
InputStream in = null;
try {
in = OpenHttpConnection(URL);
}
catch (IOException e)
{
Log.d("Networking", e.getLocalizedMessage());
return "";
}
InputStreamReader isr = new InputStreamReader(in);
int charRead;
String str = "";
char[] inputBuffer = new char[BUFFER SIZE];
try {
while ((charRead = isr.read(inputBuffer))>0) {
//---convert the chars to a String---
String readString =
String.copyValueOf(inputBuffer, 0, charRead);
str += readString;
inputBuffer = new char[BUFFER SIZE];
}
in.close();
}
catch (IOException e) {
Log.d("Networking", e.getLocalizedMessage());
return "";
}
return str;
}

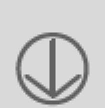
private class DownloadTextTask extends AsyncTask<String, Void, String>
{

protected String doInBackground(String... urls) {
return DownloadText(urls[0]);
}

@Override
protected void onPostExecute(String result) {
Toast.makeText(getBaseContext(), result,
Toast.LENGTH LONG).show();
}
}

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
```






```
    setContentView(R.layout.main);
    //---download text---
    new DownloadTextTask().execute(
    "http://iheartquotes.com/api/v1/random?max characters=256&max lines=10");
    }
```

然后，按 F11 键在模拟器中调试该应用。图 10-2 显示了该应用下载的网页内容。



图 10-2　通过 HTTP 下载文本内容

10.3　Socket 网络开发

虽然大部分 Web 服务都使用 HTTP 进行通信，但是它们有一个内在的缺陷：那就是这种连接是无状态的。当使用 HTTP 连接 Web 服务器时，每个连接被当作一个新的连接，Web 服务器并不会和客户端维护一个持久化的连接。

例如，在用户通过某个网站订购电影票时，当某个座位被一个用户预定后，所有其他的用户并不会立刻知晓，必须当他们再次连接到该网站服务器时才会获知该情况。这种轮询机制会占用大量的网络带宽并使应用程序效率低下。一个更好的解决方案是服务器和每个用户之间维护一个持久化的连接，每当有座位被预订时就发送消息给其他用户。

如果想要和服务器建立一个持久化的连接，并且每当变化发生时就立刻得到通知，这就需要使用一种叫做 Socket 编程的技术。我们通过一个简单的聊天程序演示如何进行 Socket 程序开发。

首先，使用 Eclipse 创建一个新的 Android 项目 Sockets。

然后，修改 AndroidManifest.xml 文件，代码如下：

```
    <?xml version="1.0" encoding="utf-8"?>
    <manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="net.learn2develop.Sockets"
```

```
android:versionCode="1"
android:versionName="1.0" >
<uses-sdk android:minSdkVersion="14" />
<uses-permission android:name="android.permission.INTERNET"/>
<application
android:icon="@drawable/ic launcher"
android:label="@string/app name" >
<activity
android:label="@string/app name"
android:name=".SocketsActivity" >
<intent-filter >
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
</application>
</manifest>
```

接着，修改 main.xml 文件，代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout height="fill parent"
android:orientation="vertical" >

<EditText
android:id="@+id/txtMessage"
android:layout width="fill parent"
android:layout height="wrap content" />

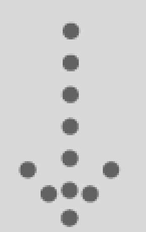
<Button
android:layout width="fill parent"
android:layout_height="wrap_content"
android:text="Send Message"
android:onClick="onClickSend"/>

<TextView
android:id="@+id/txtMessagesReceived"
android:layout width="fill parent"
android:layout height="200dp"
android:scrollbars = "vertical" />

</LinearLayout>
```

然后添加一个新的 Java 类文件 CommsThread，代码如下：

```
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.Socket;
import android.util.Log;
```



```
public class CommsThread extends Thread {

private final Socket socket;
private final InputStream inputStream;
private final OutputStream outputStream;

public CommsThread(Socket sock) {
socket = sock;
InputStream tmpIn = null;
OutputStream tmpOut = null;

try {
//---creates the inputstream and outputstream objects
// for reading and writing through the sockets---
tmpIn = socket.getInputStream();
tmpOut = socket.getOutputStream();
}
catch (IOException e) {
Log.d("SocketChat", e.getLocalizedMessage());
}

inputStream = tmpIn;
outputStream = tmpOut;
}

public void run() {
//---buffer store for the stream---
byte[] buffer = new byte[1024];
//---bytes returned from read()---
int bytes;
//---keep listening to the InputStream until an
// exception occurs---
while (true) {
try {
//---read from the inputStream---
bytes = inputStream.read(buffer);
//---update the main activity UI---
SocketsActivity.UIupdater.obtainMessage(
0,bytes, -1, buffer).sendToTarget();
}
catch (IOException e) {
break;
}
}
}

//---call this from the main activity to
// send data to the remote device---
public void write(byte[] bytes) {
try {
```

```
outputStream.write(bytes);
}
catch (IOException e) { }
}

//---call this from the main activity to
// shutdown the connection---
public void cancel() {
try {
socket.close();
} catch (IOException e) { }
}
}
```

最后，修改 SocketsActivity.java 文件，代码如下：

```
import java.io.IOException;
import java.net.InetAddress;
import java.net.Socket;
import java.net.UnknownHostException;
import android.app.Activity;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;
import android.util.Log;

public class SocketsActivity extends Activity {
static final String NICKNAME = "Wei-Meng";
InetAddress serverAddress;
Socket socket;
//---all the Views---
static TextView txtMessagesReceived;
EditText txtMessage;
//---thread for communicating on the socket---
CommsThread commsThread;

//---used for updating the UI on the main activity---
static Handler UIupdater = new Handler() {
@Override
public void handleMessage(Message msg) {
int numOfBytesReceived = msg.arg1;
byte[] buffer = (byte[]) msg.obj;
//---convert the entire byte array to string---
String strReceived = new String(buffer);
//---extract only the actual string received---
strReceived = strReceived.substring(
0, numOfBytesReceived);
```

```
//---display the text received on the TextView---
txtMessagesReceived.setText(
txtMessagesReceived.getText().toString() +
strReceived);
}
};

private class CreateCommThreadTask extends AsyncTask
<Void, Integer, Void>
{

@Override
protected Void doInBackground(Void... params) {
try {
//---create a socket---
serverAddress =
InetAddress.getByName("192.168.1.142");
//--remember to change the IP address above to match your own--
socket = new Socket(serverAddress, 500);
commsThread = new CommsThread(socket);
commsThread.start();
//---sign in for the user; sends the nick name---
sendToServer(NICKNAME);
}
catch (UnknownHostException e) {
Log.d("Sockets", e.getLocalizedMessage());
}
catch (IOException e) {
Log.d("Sockets", e.getLocalizedMessage());
}
return null;
}
}

private class WriteToServerTask extends AsyncTask
<byte[], Void, Void>
{
protected Void doInBackground(byte[]...data) {
commsThread.write(data[0]);
return null;
}
}

private class CloseSocketTask extends AsyncTask
<Void, Void, Void>
{
@Override
protected Void doInBackground(Void... params) {
try {
socket.close();
} catch (IOException e) {
```

```
Log.d("Sockets", e.getLocalizedMessage());
}
return null;
}
}

/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
//---get the views---
txtMessage = (EditText) findViewById(R.id.txtMessage);
txtMessagesReceived = (TextView)
findViewById(R.id.txtMessagesReceived);
}

public void onClickSend(View view) {
//---send the message to the server---
sendToServer(txtMessage.getText().toString());
}

private void sendToServer(String message) {
byte[] theByteArray =
message.getBytes();
new WriteToServerTask().execute(theByteArray);
}

@Override
public void onResume() {
super.onResume();
new CreateCommThreadTask().execute();
}

@Override
public void onPause() {
super.onPause();
new CloseSocketTask().execute();
}
}
```

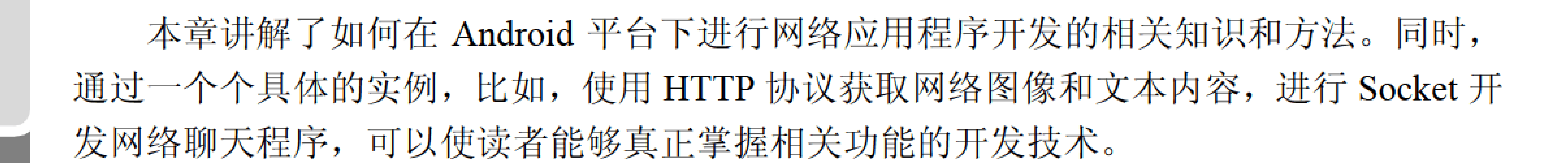
我们提供了一个 Socket 服务器程序以便测试使用。该程序是一个多用户控制台程序，在本机的 500 端口监听，把所有收到的信息发送给同它建立连接的其他客户端。打开控制台，定位到 server.exe 所在的目录，然后运行该程序，以本机的 IP 地址作为命令行参数。

确保 Android 手机和服务器在同一个网络内，然后按 F11 键启动测试该应用程序。输入消息并按 Send Message 按钮，会在服务器端收到该消息。

Socket 通信相关的功能封装到一个单独的 CommsThread 类中，该类派生自 Thread 类，使得所有 Socket 通信功能都在一个单独的线程中执行，同主 UI 线程隔离。

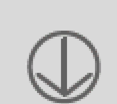
同时，还建立了 3 个 AsyncTask 类，分别是 CreateCommThreadTask、WriteToServerTask

和 CloseSocketTask，把创建 Socket 通信、向服务器发送数据、关闭 Socket 连接放到 3 个不同的异步任务中执行，确保主 UI 线程不阻塞。



10.4 本章小结

本章讲解了如何在 Android 平台下进行网络应用程序开发的相关知识和方法。同时，通过一个个具体的实例，比如，使用 HTTP 协议获取网络图像和文本内容，进行 Socket 开发网络聊天程序，可以使读者能够真正掌握相关功能的开发技术。



第 11 章　基于位置服务的应用开发

近年来，移动 APP 爆炸式增长，其中基于位置的服务 LBS（Location-Based Services）非常流行。LBS 融合了 GPS 定位、移动通信和导航等多种技术，提供了与空间位置相关的综合应用服务。LBS 发展迅速，已涉及商务、医疗、工作和生活的各个方面，为用户提供定位、追踪和敏感区域警告等一系列服务。本章主要讲解如何在 Android 应用中使用 Google Maps，以及如何通过代码操纵它。此外，还将讲解如何通过利用 Android SDK 提供的 LocationManager 功能类来获取地址位置信息。

11.1　Google Map 概述

Google Maps 作为一个独立 APP 同 Android 平台一起绑定发布，用户可以直接使用 Google Maps APP。此外，也可以把 Google Maps 嵌入到应用中并且完成一些独特的功能。本节讲述如何在自己的 Android 应用中使用 Google Maps 以及如何通过编程实现一些有用的功能。

11.1.1　显示地图

首先，使用 Eclipse 创建一个 Android 项目 lbs_demo。为了能够在创建的 Android 应用中使用 Google Maps，需要确保构建目标平台（build target）中包含 Google Maps APIs。打开项目，maps.jar 文件位于 Google APIs 文件夹内。

在集成 Google Maps 之前，需要申请一个免费的 Google Maps API key。在开发调试阶段，可以使用调试证书申请 Google Maps API key。对于 Windows 7 用户来说，调试证书位于 C:\Users\<username>\.android 目录下，文件名为 debug.keystore。申请 Google Maps API key 时，需要使用证书的 MD5 指纹。可以使用 JDK 自带的 keytool.exe 来提取证书的指纹，具体命令如下所示：

```
keytool.exe -list -alias androiddebugkey -keystore
"C:\Users\<username>\.android\debug.keystore" -storepass android
-keypass android -V
```

用浏览器打开 http://code.google.com/android/maps-api-signup.html 申请页面，输入调试证书的 MD5 指纹，按照申请要求操作，最后就可以获得 Google Maps API key 了。

接下来，打开 lbs_demo 项目的 main.xml 文件，用下面代码的粗体部分代替原来的 TextView 部分，注意，android:apiKey 属性的值要使用我们前面申请到的 API key 代替。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
```

```
android:layout width="fill parent"
android:layout height="fill parent"
android:orientation="vertical" >

<com.google.android.maps.MapView
android:id="@+id/mapView"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:enabled="true"
android:clickable="true"
android:apiKey="0AeGR0UwGH4pYmhcwaA9JF5mMEtrmwFe8RobTHA" />

</LinearLayout>
```

接着，用下面的代码代替原来的 AndroidMainfest.xml 文件。

```
    <?xml version="1.0" encoding="utf-8"?>
    <manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="net.learn2develop.LBS"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="14" />
    <uses-permission android:name="android.permission.INTERNET"/>

    <application
     android:icon="@drawable/ic launcher"
     android:label="@string/app name" >

     <uses-library android:name="com.google.android.maps" />

     <activity
       android:label="@string/app name"
       android:name=".LBSActivity" >

       <intent-filter >
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
       </intent-filter>
     </activity>
    </application>
</manifest>
```

最后，为 LBSActivity.java 文件添加显示 Google Maps 的代码部分。使用如下所示的代码替换原来的 LBSActivity.java 文件，注意，LBSActivity 应当从 MapActivity 基类派生。至此，就完成了显示 Map 的所有工作，按 F11 键在模拟器上调试应用程序。图 11-1 为通过该应用显示的一副 Google Map 地图。

```
import com.google.android.maps.MapActivity;
import android.os.Bundle;
```

```
public class LBSActivity extends MapActivity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
    }

    @Override
    protected boolean isRouteDisplayed() {
            //TODO Auto-generated method stub
            return false;
    }
}
```



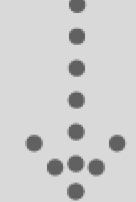
图 11-1　显示地图

为了在应用中显示 Google Maps 地图，首先需要在 manifest 文件中添加 Internet 权限，确保应用可以访问互联网。LBSActivity 类应当派生自 MapActivity 类，同时需要实现 isRouteDisplayed()方法，该方法暗示是否在地图上显示路线信息，大部分情况下，只需要简单地返回 false。

11.1.2　添加缩放控制

上节讲解了如何在自己创建的 Android 应用中显示 Google Maps 地图，本节主要介绍如何在地图上添加内置的缩放控件以使用户可以对地图进行缩放控制。

基于前面创建的项目 lbs_demo，修改 LBSActivity.java 文件，代码如下：



```
import com.google.android.maps.MapActivity;
import com.google.android.maps.MapView;
import android.os.Bundle;

public class LBSActivity extends MapActivity {

    MapView mapView;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mapView = (MapView) findViewById(R.id.mapView);
        mapView.setBuiltInZoomControls(true);
    }

    @Override
    protected boolean isRouteDisplayed() {
        //TODO Auto-generated method stub
        return false;
    }
}
```

按 F11 键在模拟器中调试应用。如图 11-2 所示，可以看到，当我们单击或者拖曳地图时，内置的缩放控件会出现在地图的底部，点击减号（-）缩小地图，点击加号（+）放大地图。



图 11-2　缩放地图

除了显示缩放控件之外，还可以通过键盘按键来缩放地图。仍然使用上面创建的项目

lbs_demo，修改 LBSActivity.java 文件，代码如下：

```
import com.google.android.maps.MapActivity;
import com.google.android.maps.MapController;
import com.google.android.maps.MapView;
import android.os.Bundle;
import android.view.KeyEvent;

public class LBSActivity extends MapActivity {
  MapView mapView;

  @Override
  public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      mapView = (MapView) findViewById(R.id.mapView);
      mapView.setBuiltInZoomControls(true);
  }

  public boolean onKeyDown(int keyCode, KeyEvent event)
  {
  MapController mc = mapView.getController();

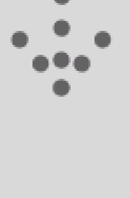
  switch (keyCode)
  {
      case KeyEvent.KEYCODE_3:
       mc.zoomIn();
       break;
      case KeyEvent.KEYCODE_1:
       mc.zoomOut();
       break;
  }
  return super.onKeyDown(keyCode, event);
}

@Override
protected boolean isRouteDisplayed() {
// TODO Auto-generated method stub
   return false;
  }
}
```

按 F11 键在模拟器中调试应用。单击数字按键 3 可以放大地图，单击数字按键 1 可以缩小地图。注意，如果把该应用部署到真实的 Android 设备上，很可能无法测试使用数字按键进行地图缩放的功能测试，因为目前大部分 Android 设备都没有配备实体键盘。

11.1.3 改变显示模式

默认情况下，Google Maps 显示在地图视图模式，用户可以使用 MapView 类的



setSatellite()方法设置 Google Maps 显示在卫星视图模式。

基于前面创建的项目 lbs_demo，修改 LBSActivity.java 文件，代码如下：

```
import com.google.android.maps.MapActivity;
import com.google.android.maps.MapController;
import com.google.android.maps.MapView;
import android.os.Bundle;
import android.view.KeyEvent;

public class LBSActivity extends MapActivity {
  MapView mapView;

  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    mapView = (MapView) findViewById(R.id.mapView);
    mapView.setBuiltInZoomControls(true);
    mapView.setSatellite(true);
  }

  public boolean onKeyDown(int keyCode, KeyEvent event)
  {
    MapController mc = mapView.getController();

    switch (keyCode)
    {
        case KeyEvent.KEYCODE_3:
          mc.zoomIn();
          break;
        case KeyEvent.KEYCODE_1:
          mc.zoomOut();
          break;
    }
    return super.onKeyDown(keyCode, event);
  }

  @Override
  protected boolean isRouteDisplayed() {
  // TODO Auto-generated method stub
    return false;
  }
}
```

图 11-3 显示了卫星视图模式下的 Google Maps。

如果还想在地图上显示当前的交通流量情况，可以使用 setTraffic()方法，基于前面创建的 lbs_demo 项目，修改 LBSActivity 类的 onCreate()方法，代码如下：

```
    @Override
    public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);
      mapView = (MapView) findViewById(R.id.mapView);
      mapView.setBuiltInZoomControls(true);
      mapView.setSatellite(true);
      mapView.setTraffic(true);
    }
```

图 11-4 是带有当前交通流量信息的 Google Maps 地图。不同颜色反映不同的流量状况。一般来说，绿色表示交通顺畅，黄色代表交通正常，红色表示交通拥堵。

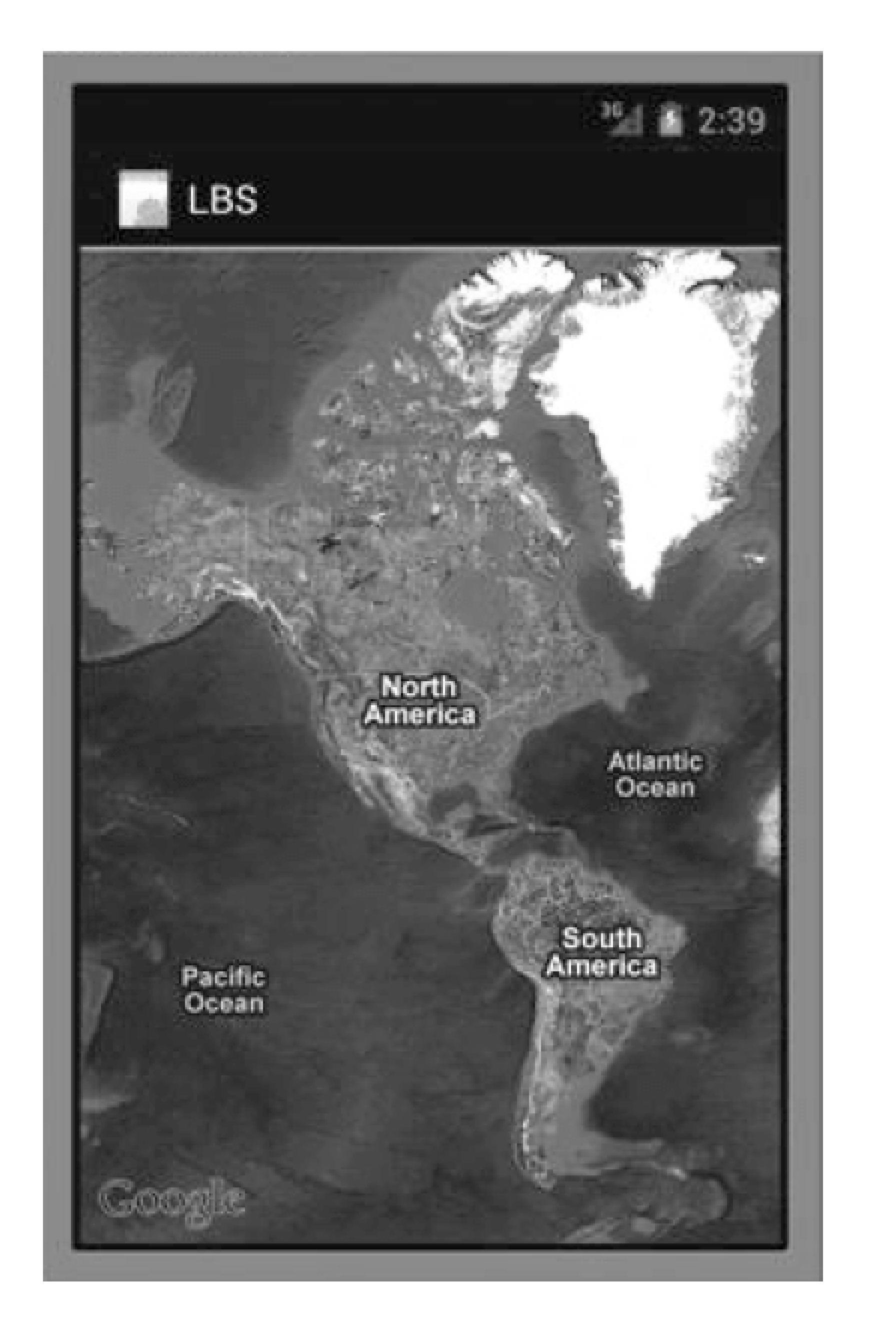


图 11-3　显示卫星地图



图 11-4　显示交通流量

11.1.4　导航到特定位置

默认情况下，Google Maps 在首次加载时显示美国地图。用户可以设置 Google Maps 显示其他特定位置的地图。

基于前面创建的项目 lbs_demo，修改 LBSActivity.java 文件，代码如下：

```
    import com.google.android.maps.GeoPoint;
    import com.google.android.maps.MapActivity;
    import com.google.android.maps.MapController;
    import com.google.android.maps.MapView;
    import android.os.Bundle;
    import android.view.KeyEvent;
```

```
public class LBSActivity extends MapActivity {
  MapView mapView;
  MapController mc;
  GeoPoint p;

  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    mapView = (MapView) findViewById(R.id.mapView);
    mapView.setBuiltInZoomControls(true);
    mapView.setSatellite(true);
    mapView.setTraffic(true);
    mc = mapView.getController();
    String coordinates[] = {"1.352566007", "103.78921587"};
    double lat = Double.parseDouble(coordinates[0]);
    double lng = Double.parseDouble(coordinates[1]);
    p = new GeoPoint(
        (int) (lat * 1E6),
        (int) (lng * 1E6));
    mc.animateTo(p);
    mc.setZoom(13);
    mapView.invalidate();
  }

  public boolean onKeyDown(int keyCode, KeyEvent event)
  {
        //...
  }

  @Override
  protected boolean isRouteDisplayed() {
        //...
  }
}
```

按 F11 键在模拟器中调试应用。当地图加载时，它动态定位到了程序中设定的坐标位置。

为了导航到一个特定的位置，我们可以使用 MapController 类的 animateTo()方法。setZoom()方法可以设定显示地图的缩放等级，数值越大，地图显示越精细。invalidate()方法强制 MapView 刷新显示。注意，GeoPoint 对象表示地理位置，它表示位置的经度和维度的度量单位是微度。例如，经度 49.897679，用 GeoPoint 表示的话，需要把它乘以 1e6 得到 49897679，然后再传给 GeoPoint 对象。

11.1.5　添加地点标记

在地图上添加地点标记可以帮助用户方便地找到目的地。本小节介绍如何在 Google

Maps 上添加地点标记。

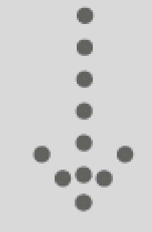
首先，把标记图像复制到上面所建项目的 res/drawable-mdpi 目录下，为了达到最好的效果，要把标记图像设置成背景透明，以便使标记图像不会遮挡地图。

然后，更改 LBSActivity.java 文件，代码如下：

```
import java.util.List;
import com.google.android.maps.GeoPoint;
import com.google.android.maps.MapActivity;
import com.google.android.maps.MapController;
import com.google.android.maps.MapView;
import com.google.android.maps.Overlay;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Point;
import android.os.Bundle;
import android.view.KeyEvent;

public class LBSActivity extends MapActivity {
  MapView mapView;
  MapController mc;
  GeoPoint p;
  private class MapOverlay extends com.google.android.maps.Overlay
  {
    @Override
    public boolean draw(Canvas canvas, MapView mapView,
      boolean shadow, long when)
    {
      super.draw(canvas, mapView, shadow);
      //转换 GeoPoint 为屏幕像素
      Point screenPts = new Point();
      mapView.getProjection().toPixels(p, screenPts);
      //添加标记
      Bitmap bmp = BitmapFactory.decodeResource(
        getResources(), R.drawable.pushpin);
      canvas.drawBitmap(bmp, screenPts.x, screenPts.y-50, null);
      return true;
  }
}

  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    mapView = (MapView) findViewById(R.id.mapView);
    mapView.setBuiltInZoomControls(true);
    mapView.setSatellite(true);
    mapView.setTraffic(true);
    mc = mapView.getController();
    String coordinates[] = {"1.352566007", "103.78921587"};
```



```
    double lat = Double.parseDouble(coordinates[0]);
    double lng = Double.parseDouble(coordinates[1]);
    p = new GeoPoint(
     (int) (lat * 1E6),
     (int) (lng * 1E6));
    mc.animateTo(p);
    mc.setZoom(13);
    //添加地点标记
    MapOverlay mapOverlay = new MapOverlay();
    List<Overlay> listOfOverlays = mapView.getOverlays();
    listOfOverlays.clear();
    listOfOverlays.add(mapOverlay);
    mapView.invalidate();
  }

  public boolean onKeyDown(int keyCode, KeyEvent event)
  {
    //...
  }

  @Override
  protected boolean isRouteDisplayed() {
    //...
  }
}
```

最后，按 F11 键在模拟器中调试该应用。图 11-5 显示添加到地图中的地点标记。



图 11-5　添加标记

11.1.6　获取地点的坐标

本小节讲解如何获取用户在屏幕上的触摸点所对应的地理位置的经纬度值。首先还是要把 Overlay 对象添加到地图中，然后需要重载 MapOverlay 类的 onTouchEvent()方法。每次用户触摸地图时都会触发执行该方法。该方法有两个参数：MotionEvent 和 MapView。使用 MotionEvent 参数的 getAction()方法，可以确定用户是否从屏幕上移开了手指。下面的代码段实现的功能是：如果用户在屏幕上触摸并移开手指，则显示触摸点对应的地理位置的经纬度值。

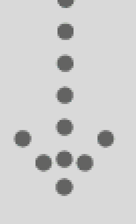
```
import java.util.List;
import com.google.android.maps.GeoPoint;

import com.google.android.maps.MapActivity;
import com.google.android.maps.MapController;
import com.google.android.maps.MapView;
import com.google.android.maps.Overlay;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Point;
import android.os.Bundle;
import android.view.KeyEvent;
import android.view.MotionEvent;
import android.widget.Toast;

public class LBSActivity extends MapActivity {
  MapView mapView;
  MapController mc;
  GeoPoint p;
  private class MapOverlay extends com.google.android.maps.Overlay
  {

    @Override
    public boolean draw(Canvas canvas, MapView mapView,
      boolean shadow, long when)
    {
      //...
    }

    @Override
    public boolean onTouchEvent(MotionEvent event, MapView mapView)
    {
        //当用户移开手指时
        if (event.getAction() == 1) {
          GeoPoint p = mapView.getProjection().fromPixels(
             (int) event.getX(),
             (int) event.getY());
```



```
            Toast.makeText(getBaseContext(),
              "Location: "+
              p.getLatitudeE6() / 1E6 + "," +
              p.getLongitudeE6() /1E6 ,
              Toast.LENGTH SHORT).show();
          }
          return false;
        }
      }
      //...
    }
```

getProjection()方法返回屏幕像素坐标和经纬度坐标之间的映射关系。fromPixels()方法把屏幕坐标转换为 GeoPoint 对象。

11.1.7　地理编码和反编码

上节中讲解了如何获得屏幕触摸点的地理位置的经纬坐标，本节主要介绍如何在地理位置的经纬坐标和实际的街道地址之间做转换。Google Maps 通过 Geocoder 类来完成这种转换。下面的代码显示如何通过 Geocoder 类的 getFromLocation()方法获取某个位置点的实际街道地址。

```
import java.io.IOException;
import java.util.List;
import java.util.Locale;
import com.google.android.maps.GeoPoint;
import com.google.android.maps.MapActivity;
import com.google.android.maps.MapController;
import com.google.android.maps.MapView;
import com.google.android.maps.Overlay;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;

import android.graphics.Point;
import android.location.Address;
import android.location.Geocoder;
import android.os.Bundle;
import android.view.KeyEvent;
import android.view.MotionEvent;
import android.widget.Toast;

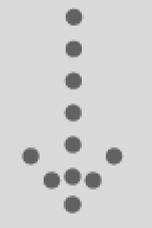
public class LBSActivity extends MapActivity {
  MapView mapView;
  MapController mc;
  GeoPoint p;

  private class MapOverlay extends com.google.android.maps.Overlay
  {
```

```
    @Override
    public boolean draw(Canvas canvas, MapView mapView,
      boolean shadow, long when)
    {
      //...
    }

    @Override
    public boolean onTouchEvent(MotionEvent event, MapView mapView)
    {
      //当用户移开手指时
      if (event.getAction() == 1) {
        GeoPoint p = mapView.getProjection().fromPixels(
          (int) event.getX(),
          (int) event.getY());
        /*
        Toast.makeText(getBaseContext(),
          "Location: "+
          p.getLatitudeE6() / 1E6 + "," +
          p.getLongitudeE6() /1E6 ,
          Toast.LENGTH SHORT).show();
        */

        Geocoder geoCoder = new Geocoder(
          getBaseContext(), Locale.getDefault());
        try {
          List<Address> addresses = geoCoder.getFromLocation(
            p.getLatitudeE6() / 1E6,
            p.getLongitudeE6() / 1E6, 1);
          String add = "";
          if (addresses.size() > 0)
          {
            for (int i=0; i<addresses.get(0).getMaxAddressLineIndex();
                 i++)
              add += addresses.get(0).getAddressLine(i) + "\n";
          }
          Toast.makeText(getBaseContext(), add,
            Toast.LENGTH SHORT).show();
        }
        catch (IOException e) {
          e.printStackTrace();
        }
        return true;
      }
      return false;
    }
  }
  //...
}
```



Geocoder 对象的 getFromLocation 方法()负责转换位置的经纬度坐标为实际的街道地址，获得街道地址后，使用 Toast 类显示出来。图 11-6 显示了地理位置的实际街道地址。



图 11-6　地理编码

反之，如果知道一个位置的街道地址，想要获取它的经纬度坐标，也可以通过 Geocoder 类来实现。下面的代码演示如何通过 Geocoder 类的 getFromLocationName()方法获取纽约帝国大厦的精确经纬度坐标。

```
@Override
public void onCreate(Bundle savedInstanceState) {

  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);
  mapView = (MapView) findViewById(R.id.mapView);
  mapView.setBuiltInZoomControls(true);
  mapView.setSatellite(true);
  mapView.setTraffic(true);
  mc = mapView.getController();

  /*
  String coordinates[] = {"1.352566007", "103.78921587"};
  double lat = Double.parseDouble(coordinates[0]);
  double lng = Double.parseDouble(coordinates[1]);
  p = new GeoPoint(
     (int) (lat * 1E6),
     (int) (lng * 1E6));
  mc.animateTo(p);
  mc.setZoom(13);
  */
```

```
  //地理位置编码
  Geocoder geoCoder = new Geocoder(this, Locale.getDefault());
  try {
    List<Address> addresses = geoCoder.getFromLocationName(
      "empire state building", 5);
    if (addresses.size() > 0) {
      p = new GeoPoint(
         (int) (addresses.get(0).getLatitude() * 1E6),
         (int) (addresses.get(0).getLongitude() * 1E6));
      mc.animateTo(p);
      mc.setZoom(20);
    }
  } catch (IOException e) {
    e.printStackTrace();
  }

  MapOverlay mapOverlay = new MapOverlay();
  List<Overlay> listOfOverlays = mapView.getOverlays();
  listOfOverlays.clear();
  listOfOverlays.add(mapOverlay);

  mapView.invalidate();
}
```

图 11-7 显示了导航到帝国大厦的地图界面。

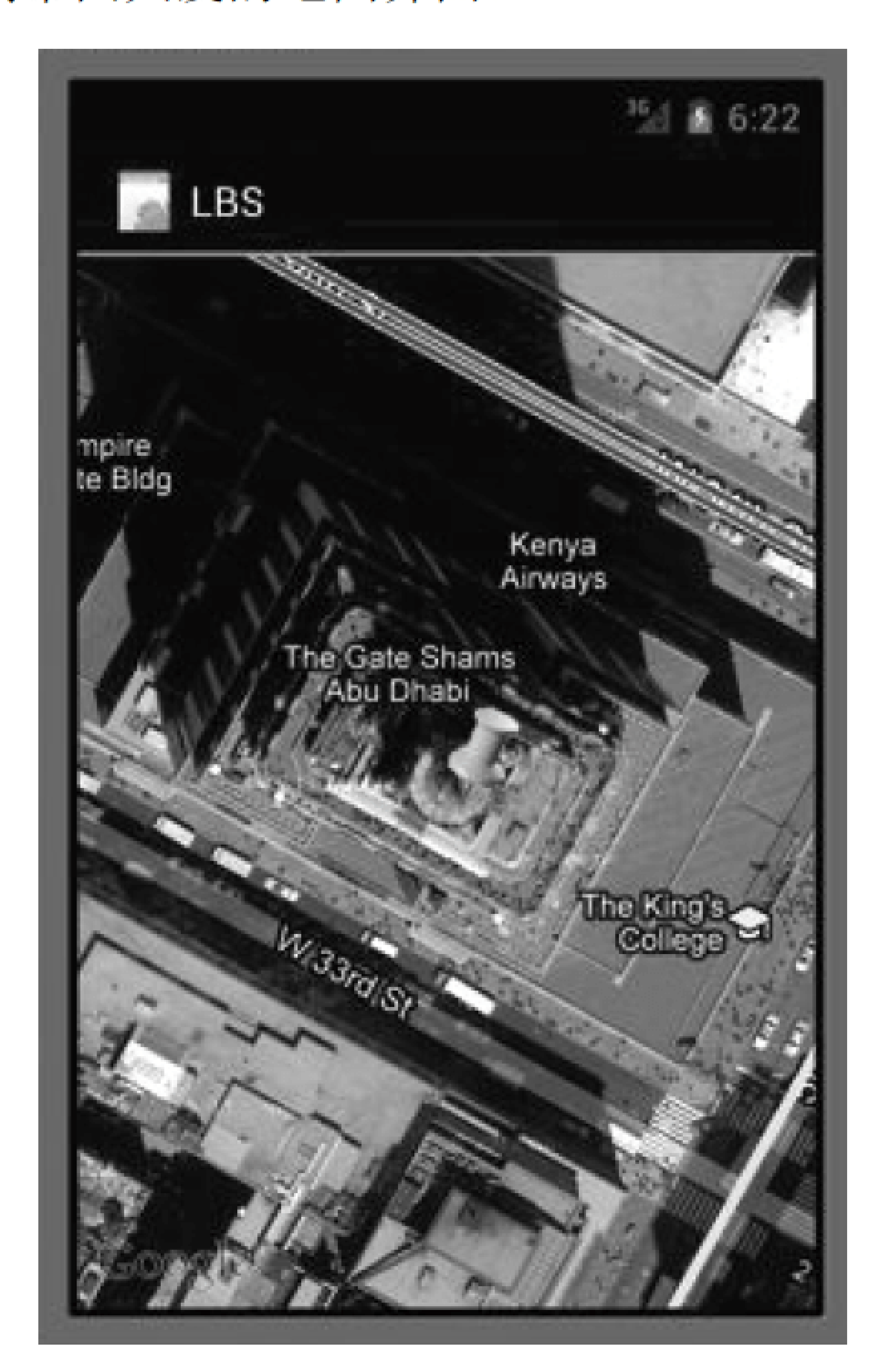
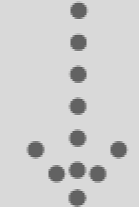


图 11-7　地理反编码



11.2 获取定位数据

目前，大多数 Android 设备都配备 GPS 接收器，可以很容易地定位用户当前所在位置。可是，GPS 必须在室外才能获得定位数据，在室内时它无法成功定位。另一种定位的方法是通过手机信号发射塔的三角测量。该方法的优点是可以在室内工作，缺点是精度不够。第三种定位方法是通过 WiFi 网络。这三种方法中，WiFi 网络的精度最低。

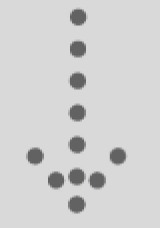
在 Android 平台上，SDK 提供 LocationManager 类来帮助设备获取位置信息。下面通过实例讲解利用 GPS 获取用户位置信息的功能。

首先，仍然使用前面创建的 lbs_demo 项目，修改 LBSActivity.java 文件，代码如下：

```
import java.io.IOException;
import java.util.List;
import java.util.Locale;
import com.google.android.maps.GeoPoint;
import com.google.android.maps.MapActivity;
import com.google.android.maps.MapController;
import com.google.android.maps.MapView;
import com.google.android.maps.Overlay;
import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.graphics.Point;
import android.location.Address;
import android.location.Geocoder;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.os.Bundle;
import android.view.KeyEvent;
import android.view.MotionEvent;
import android.widget.Toast;

public class LBSActivity extends MapActivity {
  MapView mapView;
  MapController mc;
  GeoPoint p;
  LocationManager lm;
  LocationListener locationListener;
  private class MapOverlay extends
              com.google.android.maps.Overlay
  {
    //...
  }
```

```
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mapView = (MapView) findViewById(R.id.mapView);
        mapView.setBuiltInZoomControls(true);
        mapView.setSatellite(true);
        mapView.setTraffic(true);
        mc = mapView.getController();
        /*
        String coordinates[] = {"1.352566007", "103.78921587"};
        double lat = Double.parseDouble(coordinates[0]);
        double lng = Double.parseDouble(coordinates[1]);
        p = new GeoPoint(
            (int) (lat * 1E6),
            (int) (lng * 1E6));
        mc.animateTo(p);
        mc.setZoom(13);
        */
        //---geo-coding---
        Geocoder geoCoder = new Geocoder(this, Locale.getDefault());
        try {
          //...
        }
        //---Add a location marker---
        MapOverlay mapOverlay = new MapOverlay();
        List<Overlay> listOfOverlays = mapView.getOverlays();
        listOfOverlays.clear();
        listOfOverlays.add(mapOverlay);
        mapView.invalidate();
        //---use the LocationManager class to obtain locations data---
        lm = (LocationManager)
        getSystemService(Context.LOCATION SERVICE);
        locationListener = new MyLocationListener();
    }
    @Override
    public void onResume() {
        super.onResume();
        //---request for location updates---
        lm.requestLocationUpdates(
            LocationManager.GPS PROVIDER,
            0,
            0,
            locationListener);
    }
    @Override
    public void onPause() {
        super.onPause();
        //---remove the location listener---
        lm.removeUpdates(locationListener);
    }
```



```
private class MyLocationListener implements LocationListener
{
  public void onLocationChanged(Location loc) {
  if (loc != null) {
    Toast.makeText(getBaseContext(),
      "Location changed : Lat: " + loc.getLatitude() +
      "Lng: " + loc.getLongitude(),
      Toast.LENGTH_SHORT).show();
    p = new GeoPoint(
      (int) (loc.getLatitude() * 1E6),
      (int) (loc.getLongitude() * 1E6));
    mc.animateTo(p);
    mc.setZoom(18);
  }
  }
  public void onProviderDisabled(String provider) {
  }
  public void onProviderEnabled(String provider) {
  }
  public void onStatusChanged(String provider, int status,
  Bundle extras) {
  }
}
public boolean onKeyDown(int keyCode, KeyEvent event)
{
  //...
}
@Override
protected boolean isRouteDisplayed() {
  //...
}
}
```

然后，修改 AndroidManifest.xml 文件，代码如下：

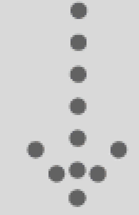
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/
android"
package="net.learn2develop.LBS"
android:versionCode="1"
android:versionName="1.0" >
  <uses-sdk android:minSdkVersion="14" />
  <uses-permission android:name="android.permission.INTERNET"/>
  <uses-permission android:name="android.permission.ACCESS_FINE_
  LOCATION"/>
  <application
android:icon="@drawable/ic_launcher"
android:label="@string/app_name" >
  <uses-library android:name="com.google.android.maps" />
  <activity
android:label="@string/app_name"
```

```
            android:name=".LBSActivity" >
                <intent-filter >
                  <action android:name="android.intent.action.MAIN" />
                  <category
                  android:name="android.intent.category.LAUNCHER" />
                </intent-filter>
              </activity>
            </application>
          </manifest>
```

最后，按 F11 键在模拟器中调试应用。注意，为了能在模拟器中模拟 GPS 数据，需要使用 DDMS 中的 Location Controls 工具，在 Location Controls 工具中输入经纬度发送出去。发送位置数据后，地图导航到新的位置。

11.3　本章小结

Android 系统作为新一代智能手机平台，在开发框架中提供了对位置服务的系统级支持，使开发人员可以使用 Java 语言很方便地开发此类应用。本章主要介绍 Android 系统中基于位置服务的应用开发，首先对 Google Maps API 的相关知识进行概述，然后详细介绍使用 Google Map API 进行程序开发的方法。



第 12 章　Android 桌面组件开发

初次启动 Android 模拟器时，可以看到在桌面上有很多图标，单击这些图标，系统就会执行相应的程序，与 Windows 操作系统桌面上的快捷方式类似，但是它不完全是快捷方式，还包括了实时文件夹（Live Folder）和桌面组件（Widget）等，这些桌面应用既美观又方便用户操作。本章将学习桌面组件的开发，让新开发的应用程序也能够出现在桌面上。

12.1　桌面快捷方式

如果系统中已经存在某一应用程序，我们可以通过生成桌面快捷方式使该应用出现在桌面上。最简单的生成快捷方式的办法是单击 menu 键（或长按桌面），之后会弹出添加桌面组件的菜单，如图 12-1 所示，进入快捷方式后，选择相应的程序即可将其添加至桌面。



图 12-1　添加程序至桌面的功能菜单

另一种办法是在编写应用程序的时候直接在程序中创建 Intent，并利用广播（Broadcast）的方式通知桌面启动器（Launcher）创建快捷方式。

然而，想要给 BroadcastReceiver 发送广播信息，必须首先具备 com.android.launcher.permission.INSTALL_SHORTCUT 权限。因此，需要在创建的 AndroidMainfest.xml 文件中声明该权限。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
     package="com.studio.android.ch10.ex1"
     android:versionCode="1"
     android:versionName="1.0">
   <application android:icon="@drawable/ji" android:label="@string/app
   name">
      <activity android:name=".UrgentCall"
```

```
                    android:label="@string/app_name">
               <intent-filter>
                   <action android:name="android.intent.action.MAIN" />
                   <category android:name="android.intent.category.LAUNCHER" />
               </intent-filter>
           </activity>
       </application>
     <uses-sdk android:minSdkVersion="3" />
    //声明权限
     <uses-permission
android:name="com.android.launcher.permission.INSTALL_SHORTCUT"/>
    </manifest>
```

接下来就需要在程序文件中创建 Intent 并广播给启动器。为了便于编写代码，下面以紧急电话为例，通过代码在桌面创建其快捷方式，代码如下：

```
    import android.app.Activity;
    import android.content.Intent;
    import android.net.Uri;
    import android.os.Bundle;
    import android.os.Parcelable;
    import android.view.View;
    import android.view.View.OnClickListener;
    import android.widget.Button;

    public class UrgentCall extends Activity implements
        OnClickListener {

        Button police;       //报警
        Button fire;         //火警
        Intent directCall;
        private final String ACTION_ADD_SHORTCUT =
            "com.android.launcher.action.INSTALL_SHORTCUT";

        @Override
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);

            setContentView(R.layout.main);

            police = (Button)findViewById(R.id.police);
            fire = (Button)findViewById(R.id.firepolice);

            police.setOnClickListener(this);
            fire.setOnClickListener(this);

            directCall = new Intent(Intent.ACTION_CALL);
        }

        @Override
        public void onClick(View v) {
```

```
            Intent addShortcut =
                new Intent(ACTION ADD SHORTCUT);
            String numToDial = null;
            Parcelable icon = null;
            switch (v.getId()) {
            case R.id.police:
                numToDial = "110";
                icon = Intent.ShortcutIconResource.fromContext(this,R.drawable.
                jing);
                break;
            case R.id.firepolice:
                numToDial = "119";
                icon = Intent.ShortcutIconResource.fromContext(this,R.drawable.
                huo);
                break;
            default:
                break;
            }
            addShortcut.putExtra(Intent.EXTRA SHORTCUT NAME,
                numToDial);
            directCall.setData(Uri.parse("tel://"+numToDial));
            addShortcut.putExtra(Intent.EXTRA SHORTCUT INTENT,
                    directCall);
            addShortcut.putExtra(Intent.EXTRA SHORTCUT ICON RESOURCE,
                    icon);
            sendBroadcast(addShortcut);
        }
    }
```

12.2 桌面组件——Widget

Widget（微块）是一种桌面上的应用程序小组件，它最初的概念是在 1998 年由一个叫 Rose 的工程师提出，但是直到 2003 年才正式为大家所知，不过随后很多大公司都开始接受并采用这一思路。

从 Android 1.5 开始，系统引入了 AppWidget 框架，开发者可以利用该框架开发 Widget，这些 Widget 可以拖到用户的桌面并且可以交互，并可让用户在主屏幕界面及时了解程序显示的重要信息。实际上，标准的 Android 系统也包含若干 Widget 的示例，如模拟时钟、音乐播放器等。

12.2.1 AppWidget 框架类

AppWidget 的框架类主要包括以下四个部分。

- **AppWidgetProvider** 继承自 BroadcastRecevier，用于在 AppWidget 发生 update、

enable、disable 和 delete 事件时接收通知。其中，onUpdate 和 onReceive 是最常用到的方法。

- ❑ **AppWidgetProviderInfo** 一般以以 XML 文件形式存在于应用的 res/xml/目录下，用于描述 AppWidget 的大小、更新频率和初始界面等信息。
- ❑ **AppWidgetManger** 用于管理 AppWidget，并向 AppWidgetProvider 发送通知。
- ❑ **RemoteViews** 能够在其他应用进程中运行的类，并向 AppWidgetProvider 发送通知。

其中，AppWidgetProvider 类的主要方法如下。

```
onDisabled(Context context)                              //禁用
onEnabled(Context context)                               //启用
onUpdate(Context  context,  AppWidgetManager  appWidgetManager,  int[]
appWidgetIds)                                            //更新
onReceive(Context context, Intent intent)                //接收
onDeleted(Context context, int[] appWidgetIds)           //删除
```

在此需要注意，由于 AppWidgetProvider 继承自 BroadcastReceiver，因此可以重载 onRecevier 方法，前提是需要在后台注册 Receiver。

AppWidgetManger 类可实现的主要方法如下。

- ❑ **bindAppWidgetId(int appWidgetId, ComponentName provider)** 通过给定的 ComponentName 绑定 appWidgetId。
- ❑ **getAppWidgetIds(ComponentName provider)** 通过给定的 ComponentName 获取 AppWidgetId。
- ❑ **getAppWidgetInfo(int appWidgetId)** 通过 AppWidgetId 获取 AppWidget 信息。
- ❑ **getInstalledProviders()** 返回一个 List<AppWidgetProviderInfo>的信息。
- ❑ **getInstance(Context context)** 获取 AppWidgetManger 实例使用的上下文对象。
- ❑ **updateAppWidget(int[] appWidgetIds, RemoteViews views)** 通过 appWidgetId 对传进来的 RemoteView 进行修改，并刷新 AppWidget 组件。
- ❑ **updateAppWidget(ComponentName provider, RemoteViews views)** 通过 ComponentName 对传入的 RemoetView 进行修改，并重新刷新 AppWidget 组件。
- ❑ **updateAppWidget(int appWidgetId, RemoteViews views)** 通过 appWidgetId 对传入的 RemoteView 进行修改，同时刷新 AppWidget 组件。

12.2.2 App Widget 的简单例子——Hello App Widget

下面用一个最简单的例子 HelloAppWidget 进行说明，然后再针对该例子具体讲解 App Widget 的技术实现。

（1）首先，新建一个名为 HelloAppWidget 的项目，注意创建时可以不选 Create Activity，如图 12-2 所示。

（2）在 layout/文件夹下创建一个 Widget 的显示布局文件 widget.xml，代码如下。



图 12-2　新建项目

```
    <?xml version="1.0" encoding="utf-8"?>
    <linearlayout android:layout height="fill parent" android:layout width=
"fill parent" android:orientation="vertical" xmlns:android="http://schemas.
    android.com/apk/res/android" android:gravity="center">
    <textview android:layout height="wrap content" android:layout width="wrap
content" android:id="@+id/textView1" android:text="欢迎进入 Widget 的世界！"
android:textcolor="#ff0000ff">
    </textview></linearlayout>
```

（3）在 xml 文件夹下创建一个 Widget 的配置文件 provider_info.xml，该文件规定了 Widget 可以占用的屏幕长宽、更新频率、所显示的布局文件（即 layout/widget.xml）等，其代码如下。

```
    <?xml version="1.0" encoding="utf-8"?>
    <!-- appwidget-provider Widget -->
    <!-- android:minWidth 最小宽度 -->
    <!-- android:minHeight 最小高度 -->
    <!-- android:updatePeriodMillis 更新频率（毫秒） -->
    <!-- android:initialLayout -->
    <!-- android:configure Widget Activity -->
    <appwidget -provider="" xmlns:android="http://schemas.android.com/apk/
res/android" android:initiallayout="@layout/widget" android:updateperiodmillis=
"86400000" android:minheight="72dp" android:minwidth="294dp">
    </appwidget>
```

（4）建立一个用于处理 Widget 请求的 Java 文件，在源文件中，HelloWidgetProvider 继承了 AppWidgetProvider 类，具体代码如下。

```
import android.appwidget.AppWidgetManager;
import android.appwidget.AppWidgetProvider;
import android.content.Context;
import android.content.Intent;
import android.util.Log;

//AppWidgetProvider 是 BroadcastReceiver 的子类
//所有请求都会发给 onReceive 方法
public class HelloWidgetProvider extends AppWidgetProvider {

//它会根据 Intent 参数中的 action 类型来决定是自己处理，还是分给下面四个特殊的方法
    @Override
    public void onReceive(Context context, Intent intent) {
        Log.i("yao", "HelloWidgetProvider --> onReceive");
        super.onReceive(context, intent);
    }

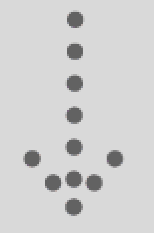
//如果 Widget 自动更新时间到了，或者有其他能够导致 Widget 变化的事件发生
//那么，onUpdate 将会被调用
    @Override
   public void onUpdate(Context context, AppWidgetManager appWidgetManager,
   int[] appWidgetIds) {
//AppWidgetManager 是 AppWidget 的管理器
//AppWidgetIds 桌面上所有的 Widget 都会被分配一个唯一的 ID 标识
        Log.i("yao", "HelloWidgetProvider --> onUpdate");
        super.onUpdate(context, appWidgetManager, appWidgetIds);
    }

//当一个 App Widget 从桌面上删除时调用
    @Override
    public void onDeleted(Context context, int[] appWidgetIds) {
        Log.i("yao", "HelloWidgetProvider --> onDeleted");
        super.onDeleted(context, appWidgetIds);
    }

//当 App Widget 第一次被放在桌面上时调用
//需要注意，同一个 App Widget 可以被放在桌面上多次
    @Override
    public void onEnabled(Context context) {
        Log.i("yao", "HelloWidgetProvider --> onEnabled");
        super.onEnabled(context);
    }

//当这个 App Widget 的最后一个实例被从桌面上移除时会调用该方法
    @Override
    public void onDisabled(Context context) {
        Log.i("yao", "HelloWidgetProvider --> onDisabled");
        super.onDisabled(context);
    }

}
```



（5）编辑 AndroidManifest.xml 文件，增加一个 receiver 标签，这个标签跟前面讲的 BroadReceiver 的配置很相似，具体代码如下。

```
    <?xml version="1.0" encoding="utf-8"?>
    <manifest     xmlns:android="http://schemas.android.com/apk/res/android"
android:versionname="1.0" android:versioncode="1" package="basic.android.
lesson35">
        <uses -sdk="" android:minsdkversion="7">

        <application android:icon="@drawable/icon" android:label="@string/
        app name">

        <!-- receiver 的 android:name 指向的是 widget 的请求接收者 -->
            <receiver android:label="Hello,App Widget" android:name=
            ".HelloWidgetProvider">
                <intent -filter="">
                    <!-- widget 默认的事件 action -->
                    <action android:name="android.appwidget.action.APPWIDGET
                    UPDATE"></action>
                </intent>
                <!-- widget 元数据，name 是固定的，resource 是 widget 的配置文件 -->
                <meta -data="" android:name="android.appwidget.provider"
                android:resource="@xml/provider info">
            </receiver>
        </application>
    </uses></manifest>
```

（6）编译并运行程序。

因为没有 main Activity，因此上述 Widget 程序即使安装完了也不会在程序列表中出现。下面讲解如何把一个 Widget 放到桌面上。

① 在模拟器中的桌面上长按，将会弹出如图 12-3 所示的对话框。

② 选择窗口小部件，如图 12-4 所示。



图 12-3　模拟器对话框



图 12-4　选择小部件

③ 选择 Hello，App Widget，如图 12-5 所示。

④ 设置成功之后，将会看到桌面上显示的一行蓝色的布局文件中的小字，效果如图 12-6 所示。



图 12-5　程序出现在窗口小部件列表中



图 12-6　最终效果

12.3　应用实例——桌面天气预报程序

本节将讲述如何开发一个可以获取天气预报信息的桌面小部件，同时实现实时异步更新界面。

开发的主要步骤如下。

（1）首先，创建 res/layout/weather_widget_layout.xml 文件用以描述部件的布局，代码如下。

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/
android"
    android:layout width="match parent"
    android:layout height="match parent"
    android:background="@drawable/weather widget back">
    <TextView
        android:layout width="wrap content"
        android:layout height="wrap content"
        android:id="@+id/city name"
        android:text="城市名称"
        android:textSize="20sp"
        android:layout alignParentLeft="true"
        android:layout marginTop="10dp"
        android:layout_marginLeft="15dp"/>
    <TextView
        android:layout width="wrap content"
        android:layout_height="wrap_content"
```

```
        android:id="@+id/cloud"
        android:text="风向"
        android:textSize="20sp"
        android:layout marginTop="10dp"
        android:layout marginRight="15dp"
        android:layout alignParentRight="true"/>
    <LinearLayout
        android:layout below="@id/city name"
        android:layout width="match parent"
        android:layout height="match parent"
        android:orientation="horizontal"
        android:layout marginBottom="10dp">
        <ImageView
            android:id="@+id/weather"
            android:layout width="0dp"
            android:layout height="match parent"
            android:src="@drawable/little_rain"
            android:layout weight="1"/>
        <TextView
            android:id="@+id/cur temp"
            android:layout width="0dp"
            android:layout height="match parent"
            android:layout weight="1"
            android:gravity="center"
            android:text="20"
            android:textSize="50sp"/>
        <LinearLayout
            android:layout_height="match_parent"
            android:layout width="0dp"
            android:layout weight="1"
            android:gravity="center"
            android:orientation="vertical">
            <TextView
                android:id="@+id/low temp"
                android:layout width="match parent"
                android:layout height="wrap content"
                android:text="最低  13"
                android:textSize="30sp"/>
            <TextView
                android:id="@+id/high temp"
                android:layout width="match parent"
                android:layout height="wrap content"
                android:text="最高  20"
                android:textSize="30sp"/>
        </LinearLayout>
    </LinearLayout>

</RelativeLayout>
```

（2）创建一个小部件的内容提供文件 res/xml/weather_info.xml，代码如下。

```
<appwidget-provider xmlns:android="http://schemas.android.com/apk/
res/android"
    android:minWidth="214dp"
    android:minHeight="142dp"
    android:updatePeriodMillis="1000"
    android:initialLayout="@layout/weather widget layout" >
 </appwidget-provider>
```

（3）创建 AppWidgetProvider 的一个子类 WeatherWidget，代码如下。

```
public class WeatherWidget extends AppWidgetProvider {
    public static WeatherFm wf = new WeatherFm();

    @Override
    public void onUpdate(Context context, AppWidgetManager appWidgetManager,
            int[] appWidgetIds) {
        super.onUpdate(context, appWidgetManager, appWidgetIds);
        Log.v("totoro","totoro1:" + appWidgetIds.length);
        LoadWeatherService.appWidgetIds = appWidgetIds;
        Log.v("totoro","totoro2:" + LoadWeatherService.appWidgetIds.length);
        context.startService(new Intent(context,LoadWeatherService.class));
    }

    public static RemoteViews updateRemoteViews(Context context) {
        RemoteViews view = new RemoteViews(context.getPackageName(),
        R.layout.weather widget layout);
        if (null == wf) {
            return null;
        } else {
            view.setTextViewText(R.id.cur temp, wf.getCurTemperature());
            view.setTextViewText(R.id.low_temp, "低" + wf.
            getLowestTemperature());
            view.setTextViewText(R.id.high_temp, "高"+
            wf.getHighestTemperature());
            return view;
        }
    }

}
```

（4）最后，在 AndroidManifest.xml 中注册。

```
<receiver
          android:name="com.monde.mondewidget.weatherwidget.WeatherWidget"
            android:icon="@drawable/ic weather widget">
            <intent-filter>
                <action android:name="android.appwidget.action.APPWIDGET
                UPDATE"/>
            </intent-filter>
            <meta-data
                android:name="android.appwidget.provider"
                android:resource="@xml/weather info"/>
</receiver>
```

以上是天气预报 widget 的基本创建步骤，如果想要实现天气的实时更新，还需要做更多的工作。如图 12-7 所示是该桌面部件所要用到的所有类，由于涉及网络访问，所以在 WeatherWidget 类的 onReceive 方法中，无法直接对 UI 进行更新。

```
com.monde.mondewidget.weatherwidget
    BDLocationUtils.java
    LoadWeatherService.java
    LocationCode.java
    LocationUtils.java
    WeatherFm.java
    WeatherQuery.java
    WeatherQueryImpl.java
    WeatherWidget.java
```

图 12-7 实例中所需的类

在第（3）步中，context.startService(new Intent(context,LoadWeatherService.class))用来启动网络访问，以便获取天气信息的服务，它的类代码如下。

```
public class LoadWeatherService extends Service implements Runnable{
    private static Object isLock = new Object();
    private static boolean isThreadRun = false;
    public static int[] appWidgetIds;

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        Log.v("onStartCommand","totoro:" + intent.toString());
        new Thread(this).start();
//      synchronized (isLock) {
//          if (!isThreadRun) {
//              isThreadRun = true;
//              new Thread(this).start();
//          }
//      }
        return super.onStartCommand(intent, flags, startId);
    }

    @Override
    public IBinder onBind(Intent intent) {
        // TODO Auto-generated method stub
        return null;
    }

    @Override
    public void run() {
        Looper.prepare();
        Log.v("onStartCommand","totoro");

        BDLocationUtils utils = new BDLocationUtils(this.
        getApplicationContext());
        utils.requestBDLocation();
```

```
            AppWidgetManager manager = AppWidgetManager.getInstance(this);
            WeatherQueryImpl impl = new WeatherQueryImpl(this);
            WeatherWidget.wf = impl.weatherQuery(utils.cityCode);
            RemoteViews view = WeatherWidget.updateRemoteViews(this);
            if (null != view) {
                manager.updateAppWidget(appWidgetIds, view);
            } else {
                Log.e("run", "更新失败");
            }

            stopSelf();
            Looper.loop();
        }
```

在该服务中，启动了一个线程去处理网络访问、获取天气信息并使用返回的数据，更新了 UI，实际的天气预报获取操作，均在 WeatherQueryImpl 中实现。

```
    public class WeatherQueryImpl implements WeatherQuery {
        private Context context;

        public WeatherQueryImpl(Context context) {
            this.context = context;
        }

        @Override
        public WeatherFm weatherQuery(String cityCode) {
            Log.v("weatherQuery","totoro:" + cityCode);
            String URL1 = "http://www.weather.com.cn/data/sk/" + cityCode +
            ".html";
            String URL2 = "http://www.weather.com.cn/data/cityinfo/" + cityCode
            + ".html";
            WeatherFm wf = new WeatherFm();
            wf.setCityCode(cityCode);
            String Weather Result = "";
            HttpGet httpRequest = new HttpGet(URL1);

            try {
                HttpClient httpClient = new DefaultHttpClient();
                HttpResponse httpResponse = httpClient.execute(httpRequest);
               if (httpResponse.getStatusLine().getStatusCode() == HttpStatus.
                SC OK) {
                    //获得返回的数据
                    Weather Result = EntityUtils.toString(httpResponse.
                    getEntity());
                    Log.v("totoro",Weather Result);
                }
            } catch (Exception e) {
                e.printStackTrace();
                return null;
            }
            //对返回数据进行解释
```




```
        if(null != Weather Result&&!"".equals(Weather Result)){
            try {
                JSONObject JO = new JSONObject(Weather Result).
                getJSONObject("weatherinfo");
                wf.setCurTemperature(JO.getString("temp"));
                wf.setCityName(JO.getString("city"));
            } catch (JSONException e) {
                e.printStackTrace();
                return null;
            }
        }

        Weather Result = "";
        httpRequest = new HttpGet(URL2);
        //获得HttpResponse对象
        try {
            HttpClient httpClient = new DefaultHttpClient();
            HttpResponse httpResponse = httpClient.execute(httpRequest);
           if (httpResponse.getStatusLine().getStatusCode() == HttpStatus.
            SC OK) {
                //获得返回数据
                Weather Result = EntityUtils.toString(httpResponse.
                getEntity());
                Log.v("totoro",Weather Result);
            }
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }
        //对返回数据进行解释
        if(null != Weather Result&&!"".equals(Weather Result)){
            try {
                JSONObject JO = new JSONObject(Weather Result).
                getJSONObject("weatherinfo");
                wf.setWeatherCondition(JO.getString("weather"));
                wf.setLowestTemperature(JO.getString("temp2"));
                wf.setHighestTemperature(JO.getString("temp1"));
            } catch (JSONException e) {
                e.printStackTrace();
                return null;
            }
        }
        return wf;
    }

    @Override
    public String cityQuery() {
        try {
            return LocationUtils.getCNByGPSlocation(context);
//          return LocationUtils.getCNByWIFILocation(context);
        } catch (Exception e) {
```

```
                e.printStackTrace();
                return null;
            }
        }

        public WeatherFm getLocalWeather () {
            return
weatherQuery(LocationCode.CHINESE LOCAL CODE.get(cityQuery()));
        }
```

天气预报的数据信息来自中国天气网 http://www.weather.com.cn/，在此使用如下接口：

http://www.weather.com.cn/data/sk/101180101.html

该接口返回的数据格式如下。

```
    "weatherinfo": {
          "city": "郑州",
          "cityid": "101180101",
          "temp": "31",
          "WD": "北风",
          "WS": "3 级",
          "SD": "58%",
          "WSE": "3",
          "time": "17:10",
          "isRadar": "1",
          "Radar": "JC RADAR AZ9371 JB"
    }
```

12.4 本章小结

本章主要阐述了桌面组件的开发，桌面应用一般可以通过两种方式来实现：一种是利用桌面快捷方式；另一种是利用 Widget 组件，这也是本章的核心。最后通过一个天气预报的例子对 Widget 组件开发进行讲解。

第 13 章　传感器应用的开发

多数安装有 Android 系统的设备都内建有多种传感器，这些传感器可以测量运动、方向以及周边的环境参数。如果想观测设备的三维运动和位置情况，或者想观测设备所处环境的温度等情况，这些传感器可以提供精确的原始测量数据。例如，一个游戏软件可能会需要跟踪设备的重力传感器获取的参数来推算用户当前的动作情况，一个旅行应用软件可能需要使用地磁场传感器和加速度传感器来提供罗盘方位信息。本章将详细讲解如何利用这些传感器的信息进行应用开发。

13.1　Android 平台传感器概述

总的来说，Android 平台支持三类传感器。

（1）运动传感器类型

这类传感器可以测量三个空间轴向上的加速度和旋转力，该类型传感器包括加速度仪、重力传感器、陀螺仪和旋转矢量传感器等。

（2）环境传感器类型

这类传感器用于测量多种环境参数，例如气温、气压、光照度和湿度，该类型传感器包括温度计、气压计和测光计等。

（3）位置传感器类型

这类传感器可以测量设备所处的物理空间的位置信息，该类型传感器包括磁力仪和方向传感器等。

我们可以通过 Android 传感器框架来获得这些传感器的原始数据。传感器框架提供了多种类和接口来辅助与传感器相关的复杂应用。Android 传感器框架可以让用户访问多种类型传感器的数据，这些传感器有些是基于硬件的，有些是基于软件的。

基于硬件的传感器是内建于手机或平板的设备的物理部件，它们直接测量环境特性来得到观测数据，例如加速度、地磁场强度或角度变化等信息。

基于软件的传感器只是通过软件来模拟硬件传感器，这类传感器在多个基于硬件的传感器的基础上通过进一步计算来达到模拟复杂传感的目的，因此也称作虚拟传感器或合成传感器。基于软件的传感器代表有方向传感器、重力传感器和线性加速度传感器。

需要注意，不同的 Android 版本会导致不同的传感器的可用性，这是因为不同的 Android 传感器的引入过程经历了不同的版本平台的更新。例如，多数传感器是在 Android 1.5 中引入的，但还有一部分一直到 Android 2.3 才被引入。

13.2　Android 传感器框架

Android 的传感器框架是 android.hardware 程序包的一部分，它包括如下类和接口。

（1）SensorManager

利用这个类可以创建一个传感器服务的实例。SensorManager 提供了多种方法，包括访问传感器数据、注册和注销传感器的事件监听器（event listener）等。SensorManager 也提供了多个传感器常量，这些常量可以用来报告传感器的精度，设置数据采样率以及对传感器进行校准等。

（2）Sensor

Sensor 类用于创建一个具体的传感器实例。该类也提供了多种方法用于确定传感器的性能。

（3）SensorEvent

系统利用 SensorEvent 来创建传感器事件对象，而该传感器事件对象可以提供如下信息：传感器的原始数据、产生该事件的传感器类型、数据的精度以及事件产生的时间戳。

（4）SensorEventListener

利用该接口创建两个回调方法，当传感器数值或精度发生改变时这两个方法用来接收相应的信息或事件。

13.3　传感器应用程序基本结构

一个典型的利用传感器信息及相关函数的应用程序一般分为两个基本工作。

（1）识别传感器和传感器性能

一些应用程序需要依赖于具体的传感器类型或性能，例如，我们可能会需要查看设备上所有的传感器，如果一些软件功能需要用到的传感器不存在，则关闭这些功能。类似的，也可以在所有类型的传感器中选择一种特定的传感器来实现，让其在应用中发挥最好的性能。

（2）监测传感器事件

监测传感器是获取传感器数据的方法。当传感器所测量的参数变化时就会发生传感器事件。如前所述，一个传感器事件提供四种信息：传感器数据、触发这个事件的传感器名、数据的精度和事件的时间戳。

13.3.1　识别传感器和传感器性能

Android 传感器框架提供了几种方法使用户很容易地确定当前运行状态下都有哪些传感器，此外，接口函数也提供了确定传感器性能的方法，诸如获取数值的最大范围、分辨率以及电源需求等。

为了识别当前设备上的传感器，首先需要得到一个传感器服务的引用。为此，需要先通过调用 getSystemService()方法创建一个 SensorManager 类的实例并传入 SENSOR_SERVICE 参数。具体参照如下代码：

```
private SensorManager mSensorManager;
...
mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
```

接下来，通过调用 getSensorList()方法，同时使用 TYPE_ALL 常量来得到当前设备上的每一个传感器的列表，例如：

```
List<Sensor> deviceSensors = mSensorManager.getSensorList(Sensor.TYPE_
ALL);
```

如果想要列出给定类型下的所有传感器，则需要使用其他常量来代替 TYPE_ALL，例如 TYPE_GYROSCOPE，TYPE_LINEAR_ACCELERATION，TYPE_GRAVITY 等。

此外，也可以通过调用 getDefaultSensor()方法，同时传入一个具体的传感器类型常量来确定某个具体类型的传感器是否存在于当前的设备上。如果一个设备上有多个同类型的传感器，则必须指派其中一个为该类型默认的传感器。如果当前类型下没有默认的传感器存在，则该方法返回 null，这意味着设备没有该类型的传感器。比如用如下代码用来检测当前设备是否有磁力仪传感器：

```
private SensorManager mSensorManager;
...
mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
if (mSensorManager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD) != null){
  //有磁力仪，并进行后续工作
  }
else {
  //没有磁力仪
  }
```

除了能够列出设备上所有的传感器外，用户还可以使用 Sensor 类的公共方法来确定某个传感器的性能和属性。如果想要让程序可以根据不同的传感器或传感器性能来自行调整，这一方法将非常有用。例如，可以使用 getResolution()和 getMaximumRange()方法来得到某个传感器的分辨率和测量的最大范围，此外还可以使用 getPower()方法来获取到传感器的电源需求。

不同的生产商会导致传感器差异较大，因此，公共方法中还有两个方法特别重要，getVendor()和 getVersion()，它们用来获取当前传感器的生产商及版本信息。如下代码将演示如何使用这两个方法，在该例子中，将要寻找生产商为 Google Inc.、版本为 3 的重力传感器。如果没有这样的传感器，系统设备则采用加速度传感器来代替。

```
private SensorManager mSensorManager;
private Sensor mSensor;

...

mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);

if (mSensorManager.getDefaultSensor(Sensor.TYPE_GRAVITY) != null){
  List<Sensor> gravSensors = mSensorManager.getSensorList(Sensor.TYPE_
  GRAVITY);
  for(int i=0; i<gravSensors.size(); i++) {
    if ((gravSensors.get(i).getVendor().contains("Google Inc.")) &&
       (gravSensors.get(i).getVersion() == 3)){
```

```
        //使用 Google 生产的版本为 3 的重力传感器
        mSensor = gravSensors.get(i);
      }
    }
  }
  else{
    //使用加速度传感器
    if (mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER) != null){
      mSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
    }
    else{
      //当前设备没有加速度传感器，程序无法进行
    }
  }
```

另一个有用的方法是 getMinDelay()，它用来返回传感器感知数据时的最小时间间隔（以微秒为单位），任何返回非零值的传感器称为流传感器，流传感器一般以一个固定的时间间隔来感知数据。相反，getMinDelay()的返回值为 0，就表示该传感器不是流式传感器，这种传感器只在被监测的参数发生变化时才会传回数据。getMinDelay()方法可以设定传感器的最大采样率，所以该方法也很实用。如果应用程序中的某些功能需要较高的采样率，我们就可以利用 getMinDelay()来确定传感器是否符合设计要求，而后再决定启用或禁用相关功能。需要注意，传感器框架不一定是按其最大采样率来向应用程序传送数据的，传感器框架是通过传感器事件来传送数据，由于传感器事件会受很多因素影响，因而，采样率会变化。

13.3.2 监测传感器事件

接下来需要实现监听传感器的原始数据，我们需要实现 SensorEventListener 接口的 onAccuracyChanged() 和 onSensorChanged() 两个回调方法。Android 系统仅当下列事件发生时调用这两个方法：

（1）当传感器的精度发生变化时

该情况下，系统将调用 onAccuracyChanged()方法，同时传入相应的 Sensor 对象的引用及新的传感器精度值。精度的表示一般包括四个状态常量：

SENSOR_STATUS_ACCURACY_LOW

SENSOR_STATUS_ACCURACY_MEDIUM

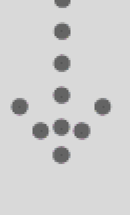
SENSOR_STATUS_ACCURACY_HIGH

SENSOR_STATUS_UNRELIABLE。

（2）当传感器返回新的数据时

该情况下，系统将调用 onSensorChanged()方法，同时传入一个 SensorEvent 对象，其中包含了有关新数据的信息，包括精度、产生新数据的传感器、新数据产生的时间戳及新数据内容。

如下代码显示了如何用 onSensorChanged()方法来控制光敏传感器，同时将感知到的原始数据显示在一个 XML 文件的 TextView 中。



```
public class SensorActivity extends Activity implements SensorEventListener {
private SensorManager mSensorManager;
private Sensor mLight;

@Override
public final void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);

  mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
  mLight = mSensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);
}

@Override
public final void onAccuracyChanged(Sensor sensor, int accuracy) {
  //如果传感器的精度发生变化则进行相应任务
}

@Override
public final void onSensorChanged(SensorEvent event) {
  //光敏传感器返回单个值。
  //多数传感器会返回 3 个值，每个值代表每个坐标轴上的信息。
  float lux = event.values[0];
  //利用该值进行后续工作
}

@Override
protected void onResume() {
  super.onResume();
  mSensorManager.registerListener(this, mLight, SensorManager.SENSOR_
  DELAY_NORMAL);
}

@Override
protected void onPause() {
  super.onPause();
  mSensorManager.unregisterListener(this);
}
}
```

在该代码中，registerListener()被调用的时候指派了一个默认的数据延迟，即 SENSOR_DELAY_NORMAL，它控制着传感器事件的触发间隔。数据延迟可以变为其他值，例如 SENSOR_DELAY_GAME（20000 微秒）、SENSOR_DELAY_UI（60000 微秒）或 SENSOR_DELAY_FASTEST（0 微秒）。Android 3.0 之后的版本可以直接设定任意数值。但是一般设定的数值只是个建议延迟时间值，系统和其他应用程序可以根据情况修改这一数值。由于系统一般使用比设定值小一些的数值，因此在设定这一数值时尽量使用较大的

数值，这样做也可以降低处理器的负载并达到减少耗电量的目的。

此外，上面例子中使用了 onResume()和 onPause()两个回调方法来实现注册和注销传感器事件侦听器。在设计程序时要注意及时关闭传感器，否则某些传感器可能耗电很大。

13.4 运动传感器

Android 提供了多种运动传感器监测设备的运动情况，这些传感器中有两个总是基于硬件的，即加速度仪和旋转矢量传感器，而重力传感器、线性加速度仪和磁力仪既可以基于软件又可以基于硬件。例如，某些设备上基于软件的传感器一般是在加速度仪和磁力仪基础上计算出相应数据，而有些甚至还需要用到陀螺仪。大多数 Android 设备都会有加速度仪和陀螺仪。而多种多样的基于软件的传感器可用性会更大一些，因为它们大多是在多个硬件传感器基础上产生数据的，可以提供复杂的数据信息。

13.4.1 运动类型传感器简介

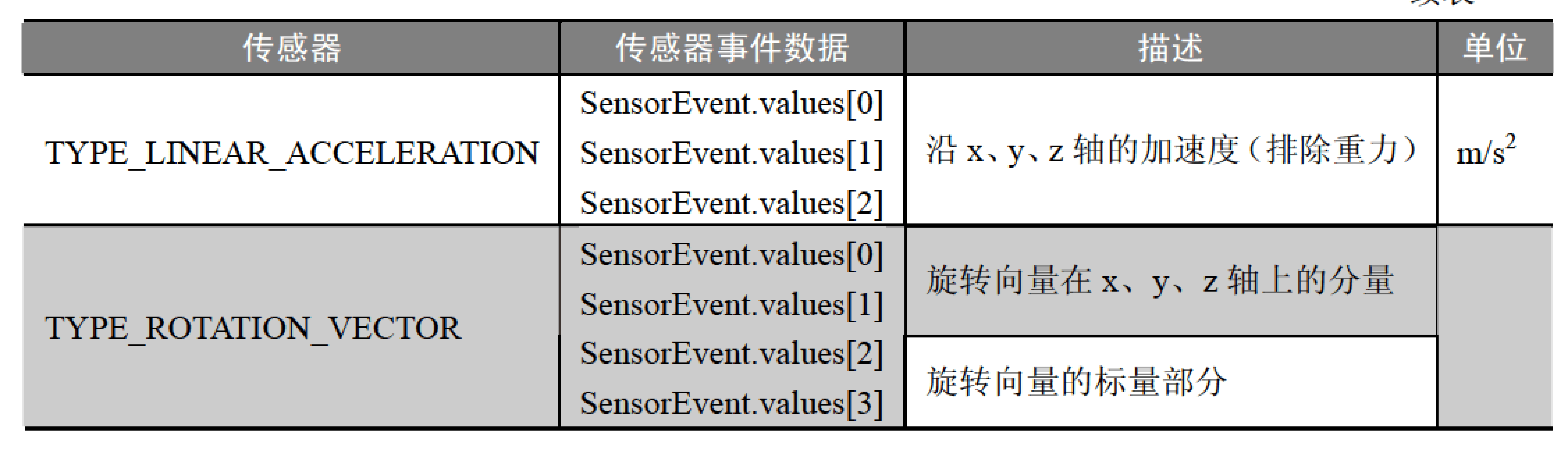
运动类型的传感器对监测设备的运动（如倾斜、晃动、旋转等）至关重要。通常晃动、倾斜、旋转等动作既可以作为用户的输入信息，又可以反映设备所处的物理环境的参数变化。所有的运动类型传感器都会在 SensorEvent 中返回由多维数组表示的传感器数据。

例如，在某个传感器事件对象中，加速度仪返回其三维坐标系上的加速度数据，而陀螺仪则返回三维坐标系上的旋转速度数据。这些数值以 float 数组的形式作为参数返回。表 13-1 为 Android 系统下常见运动传感器及其数值情况。

表 13-1　Android 系统支持的常见运动传感器

传感器	传感器事件数据	描述	单位
TYPE_ACCELEROMETER	SensorEvent.values[0] SensorEvent.values[1] SensorEvent.values[2]	沿 x、y、z 轴的加速度（包括重力）	m/s^2
TYPE_GRAVITY	SensorEvent.values[0] SensorEvent.values[1] SensorEvent.values[2]	沿 x、y、z 轴的重力加速度	m/s^2
TYPE_GYROSCOPE	SensorEvent.values[0] SensorEvent.values[1] SensorEvent.values[2]	围绕 x、y、z 轴旋转的转速	rad/s
TYPE_GYROSCOPE_UNCALIBRATED	SensorEvent.values[0] SensorEvent.values[1] SensorEvent.values[2]	围绕 x、y、z 轴旋转的转速（不包含漂移补偿）	rad/s
	SensorEvent.values[3] SensorEvent.values[4] SensorEvent.values[5]	围绕 x、y、z 轴旋转的漂移估计值	

续表



传感器	传感器事件数据	描述	单位
TYPE_LINEAR_ACCELERATION	SensorEvent.values[0] SensorEvent.values[1] SensorEvent.values[2]	沿 x、y、z 轴的加速度（排除重力）	m/s^2
TYPE_ROTATION_VECTOR	SensorEvent.values[0] SensorEvent.values[1] SensorEvent.values[2]	旋转向量在 x、y、z 轴上的分量	
	SensorEvent.values[3]	旋转向量的标量部分	

旋转矢量传感器和重力传感器在监测设备运动中经常被使用到。旋转矢量传感器经常会用在与运动有关的程序中，例如识别手势、角度的变化，监测方位的变化等，所以它在游戏开发及相机防抖等应用程序中将非常实用。

13.4.2 基本运动传感器的使用

下面对几种基本运动传感器的使用进行介绍。

1. 加速度传感器

加速度传感器可以测量设备的加速度情况，包括重力。下面的代码将展示如何获得一个缺省的加速度传感器实例。

```
private SensorManager mSensorManager;
private Sensor mSensor;
  ...
mSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
mSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
```

理论上讲，加速度传感器是通过测量施加到传感器上的作用力（Fs），然后以下列公式来推算出加速度（Ad），其中，m 表示质量：

$$A_d = -\sum F_s / m$$

但是，重力（g）方向会影响到加速度的测量，因此有如下关系：

$$A_d = -g - \sum F_s / m$$

所以，要想准确测量出加速度值，必须排除加速度数据中的重力影响。一般是通过高通滤波将重力从测得的加速度数据中去除。

```
public void onSensorChanged(SensorEvent event){
  //alpha 通过 t / (t + dT)计算得到
  //其中 t 是低通滤波器的时间常量
  //dT 是事件交付频率
  //实际使用时，alpha 可以是其他数值

  final float alpha = 0.8;
```

```
    //通过低通滤波器分离出重力加速度
    gravity[0] = alpha * gravity[0] + (1 - alpha) * event.values[0];
    gravity[1] = alpha * gravity[1] + (1 - alpha) * event.values[1];
    gravity[2] = alpha * gravity[2] + (1 - alpha) * event.values[2];

    //通过高通滤波器去除重力影响
    linear acceleration[0] = event.values[0] - gravity[0];
    linear acceleration[1] = event.values[1] - gravity[1];
    linear acceleration[2] = event.values[2] - gravity[2];
  }
```

加速度传感器使用标准的传感器坐标系，因此在实际使用中，当设备平放在平坦的桌面上时，会发生以下情况：

- ❑ 当从左侧平推设备时（此时设备向右移动），x 方向的加速度为正值；
- ❑ 当从设备底部向前平推时（此时设备将沿着远离用户的方向移动），y 方向加速度为正值；
- ❑ 当以 A m/s^2 的加速度向上方移动设备时，z 方向的加速度为 A+9.81，即设备的加速度值（+A m/s^2）减去重力加速度值（-9.81 m/s^2）;
- ❑ 当设备静止时，z 方向的加速度为+9.81。

通常情况下，加速度传感器足够满足多种运动情况的监测，而且几乎所有 Android 平台设备都带有加速度传感器。其优点是能耗比其他运动类传感器小 10 倍，缺点是为了要抵消重力的影响必须配合实现高通和低通滤波操作。

2. 重力传感器

重力传感器以一个三维向量来表示重力的方向和大小，下列代码讲述如何获得一个缺省的重力传感器。

```
private SensorManager mSensorManager;
private Sensor mSensor;
...
mSensorManager = (SensorManager) getSystemService(Context.SENSOR SERVICE);
mSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_GRAVITY);
```

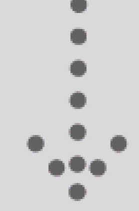
与加速度传感器一样，重力传感器的单位也是 m/s^2，坐标系也相同。同时需要注意，当设备静止时，重力传感器的输出值应当与加速度传感器的输出相同。

3. 陀螺仪

陀螺仪用于测量设备围绕三个坐标轴旋转的速率，单位为弧度/秒（rad/s）。下列代码展示了如何获得一个缺省的陀螺仪：

```
private SensorManager mSensorManager;
private Sensor mSensor;
...
mSensorManager = (SensorManager) getSystemService(Context.SENSOR SERVICE);
mSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE);
```

陀螺仪使用的坐标系与加速度传感器相同，如果从 x、y、z 轴的正向位置观测处于轴心位置的设备时，当设备逆时针方向旋转，则在相应轴向上的值为正值。陀螺仪的输出数



据表示转动弧度的变化率。

```
    private static final float NS2S = 1.0f / 1000000000.0f;
    private final float[] deltaRotationVector = new float[4]();
    private float timestamp;

    public void onSensorChanged(SensorEvent event) {
      //利用陀螺仪采样数据计算时间间隔的偏移量
      //偏移量将与当前旋转向量相乘
      if (timestamp != 0) {
        final float dT = (event.timestamp - timestamp) * NS2S;
        //没有规格化的旋转向量坐标值
        float axisX = event.values[0];
        float axisY = event.values[1];
        float axisZ = event.values[2];

        //计算旋转速度
        float omegaMagnitude = sqrt(axisX*axisX + axisY*axisY + axisZ*axisZ);

        //如果旋转向量偏移值大到能够获得坐标值时进行规格化旋转向量
        //EPSILON 是计算偏移量的初始值。当偏移量小于该值则不予计算
        if (omegaMagnitude > EPSILON) {
          axisX /= omegaMagnitude;
          axisY /= omegaMagnitude;
          axisZ /= omegaMagnitude;
        }

        //把围绕坐标轴旋转的角速度与时间间隔合并表示以便得到采样间隔的旋转偏移量
        //在转换为旋转矩阵之前需要把围绕坐标轴旋转的角度表示成四元组
        float thetaOverTwo = omegaMagnitude * dT / 2.0f;
        float sinThetaOverTwo = sin(thetaOverTwo);
        float cosThetaOverTwo = cos(thetaOverTwo);
        deltaRotationVector[0] = sinThetaOverTwo * axisX;
        deltaRotationVector[1] = sinThetaOverTwo * axisY;
        deltaRotationVector[2] = sinThetaOverTwo * axisZ;
        deltaRotationVector[3] = cosThetaOverTwo;
      }
      timestamp = event.timestamp;
      float[] deltaRotationMatrix = new float[9];
      SensorManager.getRotationMatrixFromVector(deltaRotationMatrix,
deltaRotationVector);
        //通过下式更新旋转向量。
        //rotationCurrent = rotationCurrent * deltaRotationMatrix;
       }
   }
```

在实际应用中，陀螺仪的噪声和偏移都会产生误差，这就需要一定的补偿，一般要利用其他传感器得到的信息来确定噪声和偏移的具体数值，例如重力传感器和加速度传感器等。

13.5 利用加速度仪监测设备摇动

加速度仪、陀螺仪等硬件使软件可以检测到用户在使用过程中的动作，从而使人和手机的沟通方式不再仅仅是通过键盘，通过简单的手势或动作进行交互可以提供更自然的沉浸式用户体验。摇动作为基本的动作之一，被很多软件用作主要的交互方式，例如，通过摇动来实现重置手机设置、随机选择或清除数据等功能。而对于程序设计者而言，检测摇动则需要利用加速度仪。这一小节内容将讲解如何利用加速度仪实现对摇动的检测并触发相应任务。

首先，创建一个新的 Android 工程。为了能够直观地显示当前手机是否在摇动，在/res/drawable 文件夹中加入两个指示图片，如图 13-1 所示。

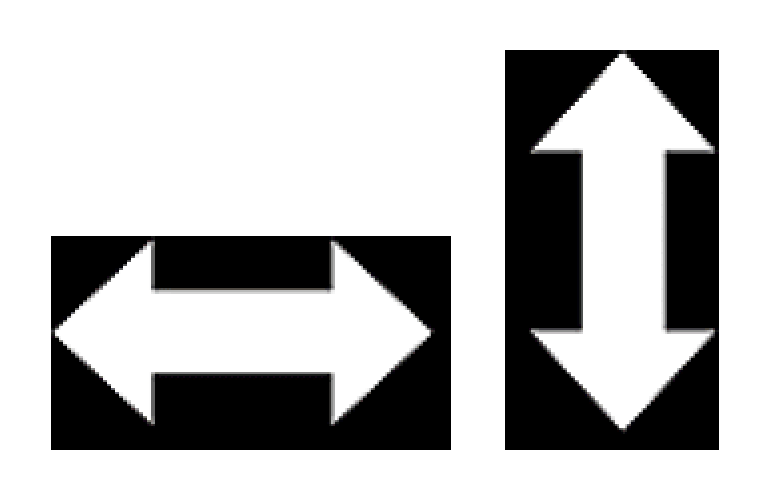


图 13-1　用于指示摇动方向的图片

然后，在/res/layout 文件夹中，添加一个名为 main.xml 的文件，整个布局包括一个两行三列的表格和一个 image view。代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout width="fill parent"
android:layout height="fill parent">
<TextView
android:paddingTop="20dip"
android:layout width="fill parent"
android:layout height="wrap content"
android:textSize="16sp"
android:textStyle="bold"
android:gravity="center"
android:text="Shaker Demo"/>
<TableLayout
android:paddingTop="10dip"
android:layout width="fill parent"
android:layout height="wrap content"
android:stretchColumns="*">
<TableRow>
<TextView
android:layout width="wrap content"
android:layout_height="wrap_content"
```

```
    android:textSize="14sp"
    android:text="X-Axis"
    android:gravity="center"/>
    <TextView
    android:layout width="wrap content"
    android:layout height="wrap content"
    android:textSize="14sp"
    android:text="Y-Axis"
    android:gravity="center"/>
    <TextView
    android:layout width="wrap content"
    android:layout height="wrap content"
    android:textSize="14sp"
    android:text="Z-Axis"
    android:gravity="center"/>
    </TableRow>
    <TableRow>
    <TextView
    android:layout width="wrap content"
    android:layout height="wrap content"
    android:id="@+id/x axis"
    android:gravity="center"/>
    <TextView
    android:layout width="wrap content"
    android:layout height="wrap content"
    android:id="@+id/y axis"
    android:gravity="center"/>
    <TextView
    android:layout width="wrap content"
    android:layout height="wrap content"
    android:id="@+id/z axis"
    android:gravity="center"/>
    </TableRow>
    </TableLayout>
    <ImageView
    android:paddingTop="10dip"
    android:layout width="wrap content"
    android:layout height="wrap content"
    android:id="@+id/image"
    android:layout gravity="center"
    android:visibility="invisible"/>
    </LinearLayout>
```

在/src 文件夹中创建一个 Main.java 文件，主要工作是继承和实现 SensorEventListener。此外，在 onCreate 方法中，还需要初始化一些变量，得到一个加速度仪的实例，同时还需要注册传感器事件监听器。代码如下：

```
    package com.author;
    import android.app.Activity;
    import android.content.Context;
    import android.hardware.Sensor;
```

```
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;
public class Main extends Activity implements SensorEventListener {
private float mLastX, mLastY, mLastZ;
private boolean mInitialized; private SensorManager mSensorManager; private
Sensor mAccelerometer; private final float NOISE = (float) 2.0;
//第一次创建时执行

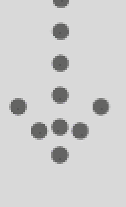
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
mInitialized = false;
mSensorManager = (SensorManager) getSystemService(Context.SENSOR SERVICE);
mAccelerometer = mSensorManager.getDefaultSensor(Sensor.TYPE ACCELEROMETER);
mSensorManager.registerListener(this, mAccelerometer, SensorManager.
SENSOR DELAY NORMAL);
}
}
```

在实现传感器事件句柄前需要重写 Main.java 中的管理对象生命周期的方法。

```
protected void onResume() {
super.onResume();
mSensorManager.registerListener(this, mAccelerometer, SensorManager.
SENSOR DELAY NORMAL);
}
protected void onPause() {
super.onPause();
mSensorManager.unregisterListener(this);
}
@Override
public void onAccuracyChanged(Sensor sensor, int accuracy) {
//在该实例中可省略
}
```

在 Main.java 中加入 onSensorChanged 句柄。如前所述，加速度仪会返回三个数值，分别对应 x、y、z 轴。其关键在于得到每个轴向上两次函数调用之间的递增量 delta。这些递增量还需要与一个常量（NOISE）进行比较并以此判断是否为噪声，以防误判。

```
@Override
public void onSensorChanged(SensorEvent event) {
TextView tvX= (TextView)findViewById(R.id.x axis);
TextView tvY= (TextView)findViewById(R.id.y axis);
TextView tvZ= (TextView)findViewById(R.id.z axis);
ImageView iv = (ImageView)findViewById(R.id.image);
```



```
float x = event.values[0];
float y = event.values[1];
float z = event.values[2];
if (!mInitialized) {
mLastX = x;
mLastY = y;
mLastZ = z;
tvX.setText("0.0");
tvY.setText("0.0");
tvZ.setText("0.0");
mInitialized = true;
} else {
float deltaX = Math.abs(mLastX - x);
float deltaY = Math.abs(mLastY - y);
float deltaZ = Math.abs(mLastZ - z);
if (deltaX < NOISE) deltaX = (float)0.0;
if (deltaY < NOISE) deltaY = (float)0.0;
if (deltaZ < NOISE) deltaZ = (float)0.0;
mLastX = x;
mLastY = y;
mLastZ = z;
tvX.setText(Float.toString(deltaX));
tvY.setText(Float.toString(deltaY));
tvZ.setText(Float.toString(deltaZ));
iv.setVisibility(View.VISIBLE);
if (deltaX > deltaY) {
iv.setImageResource(R.drawable.horizontal);
} else if (deltaY > deltaX) {
iv.setImageResource(R.drawable.vertical);
} else {
iv.setVisibility(View.INVISIBLE);
}
}
}
```

程序完成后，当用户左右晃动设备的时候就会出箭头指示，如图 13-2 所示。

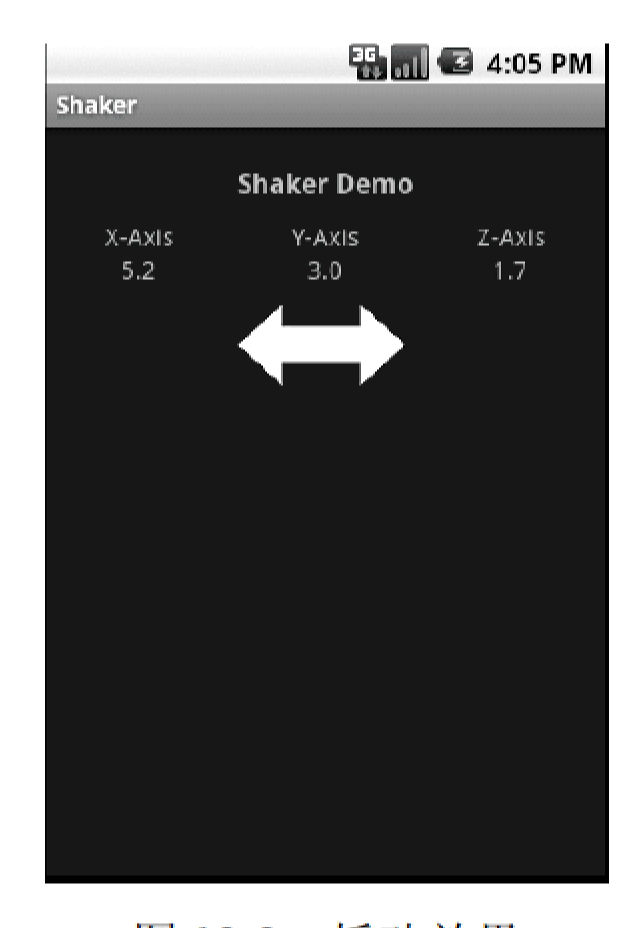


图 13-2　摇动效果

13.6 利用传感器实现指南针功能

加速度仪是基于硬件的，如前所述，其实还有很多传感器是基于软件的，例如，比较常用到的方位传感器（TYPE_ORIENTATION）就是在加速度仪和磁力仪的基础上通过软件的整合计算产生的。下面将讲述如何利用这些传感器资源编写一个简易的指南针程序。

首先创建一个 Android 工程。在 AndroidManifest.xml 文件中，设置请求精确（fine）或粗糙（coarse）定位的请求。在该例子中仍采用纵向（portrait）模式显示。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.authorwjf.whichwayisRight"
   android:versionCode="1"
   android:versionName="1.0" >

   <uses-sdk
      android:minSdkVersion="14"
      android:targetSdkVersion="19" />

   <uses-permission android:name="android.permission.ACCESS FINE
   LOCATION" />
   <uses-permission android:name="android.permission.ACCESS COARSE
   LOCATION" />

   <application
      android:allowBackup="true"
      android:icon="@drawable/ic launcher"
      android:label="@string/app name"
      android:theme="@style/AppTheme" >
      <activity
         android:screenOrientation="portrait"
         android:name=".MainActivity"
         android:label="@string/app name" >
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>
   </application>

</manifest>
```

为了能够指示方向，还需要一个如图 13-3 所示的箭头图片，将其放入 /res/drawable-xhdpi 文件夹。



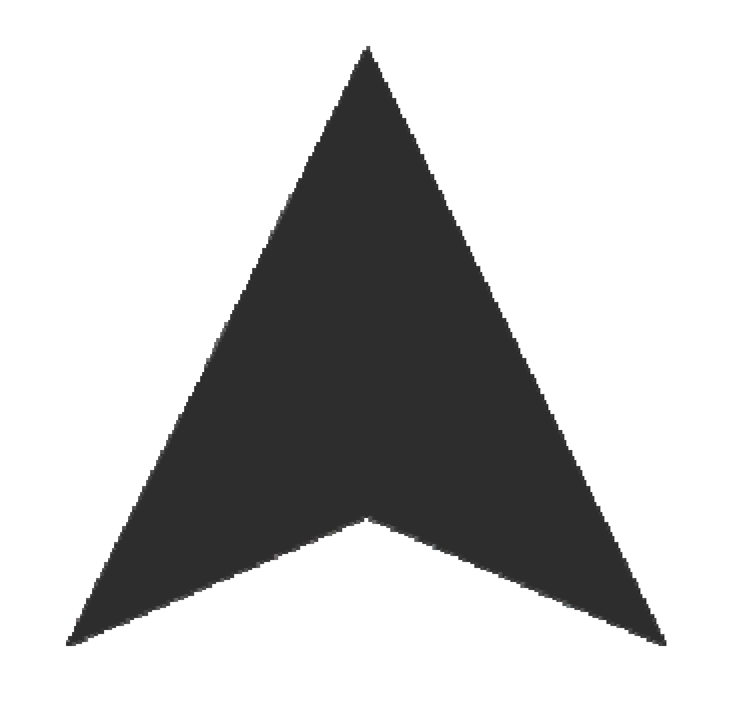


图 13-3　指南针程序需要的指示图片

在/res/layout 文件夹中创建一个名为 activity_main.xml 文件，代码如下：

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout width="match parent"
    android:layout height="match parent"
    android:paddingBottom="@dimen/activity vertical margin"
    android:paddingLeft="@dimen/activity horizontal margin"
    android:paddingRight="@dimen/activity horizontal margin"
    android:paddingTop="@dimen/activity vertical margin"
    tools:context="com.authorwjf.whichwayisright.MainActivity" >

    <ImageView
        android:id="@+id/pointer"
        android:layout width="wrap content"
        android:layout height="wrap content"
        android:layout centerInParent="true"
        android:src="@drawable/pointer" />

</RelativeLayout>
```

随后在/src 文件夹中创建一个名为 MainActivity.java 的源程序文件，在此不再细讲传感器数据的计算与转换。另外需要注意传感器的注册和注销，因为这样就不会使程序过度耗电。

```
import android.app.Activity;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.view.animation.Animation;
import android.view.animation.RotateAnimation;
import android.widget.ImageView;

public class MainActivity extends Activity implements SensorEventListener {

    private ImageView mPointer;
    private SensorManager mSensorManager;
```



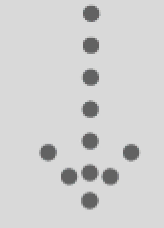
```
        private Sensor mAccelerometer;
        private Sensor mMagnetometer;
        private float[] mLastAccelerometer = new float[3];
        private float[] mLastMagnetometer = new float[3];
        private boolean mLastAccelerometerSet = false;
        private boolean mLastMagnetometerSet = false;
        private float[] mR = new float[9];
        private float[] mOrientation = new float[3];
        private float mCurrentDegree = 0f;

        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_main);
            mSensorManager = (SensorManager)getSystemService(SENSOR_SERVICE);
            mAccelerometer = mSensorManager.getDefaultSensor(Sensor.TYPE_
            ACCELEROMETER);
            mMagnetometer = mSensorManager.getDefaultSensor(Sensor.TYPE_
            MAGNETIC_FIELD);
            mPointer = (ImageView) findViewById(R.id.pointer);
        }

        protected void onResume() {
            super.onResume();
            mSensorManager.registerListener(this, mAccelerometer, SensorManager.
            SENSOR_DELAY_GAME);
            mSensorManager.registerListener(this, mMagnetometer, SensorManager.
            SENSOR_DELAY_GAME);
        }

        protected void onPause() {
            super.onPause();
            mSensorManager.unregisterListener(this, mAccelerometer);
            mSensorManager.unregisterListener(this, mMagnetometer);
        }

        @Override
        public void onSensorChanged(SensorEvent event) {
            if (event.sensor == mAccelerometer) {
                System.arraycopy(event.values, 0, mLastAccelerometer, 0, event.
                values.length);
                mLastAccelerometerSet = true;
            } else if (event.sensor == mMagnetometer) {
                System.arraycopy(event.values, 0, mLastMagnetometer, 0, event.
                values.length);
                mLastMagnetometerSet = true;
            }
            if (mLastAccelerometerSet && mLastMagnetometerSet) {
                SensorManager.getRotationMatrix(mR, null, mLastAccelerometer,
                mLastMagnetometer);
                SensorManager.getOrientation(mR, mOrientation);
                float azimuthInRadians = mOrientation[0];
                float azimuthInDegress = (float)(Math.toDegrees(azimuthInRadians)
                +360)%360;
                RotateAnimation ra = new RotateAnimation(
```



```
                    mCurrentDegree,
                     -azimuthInDegress,
                     Animation.RELATIVE TO SELF, 0.5f,
                     Animation.RELATIVE TO SELF,
                     0.5f);

            ra.setDuration(250);

            ra.setFillAfter(true);

            mPointer.startAnimation(ra);
            mCurrentDegree = -azimuthInDegress;
        }
    }

    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {
        //TODO Auto-generated method

    }

}
```

运行效果如图 13-4 所示。

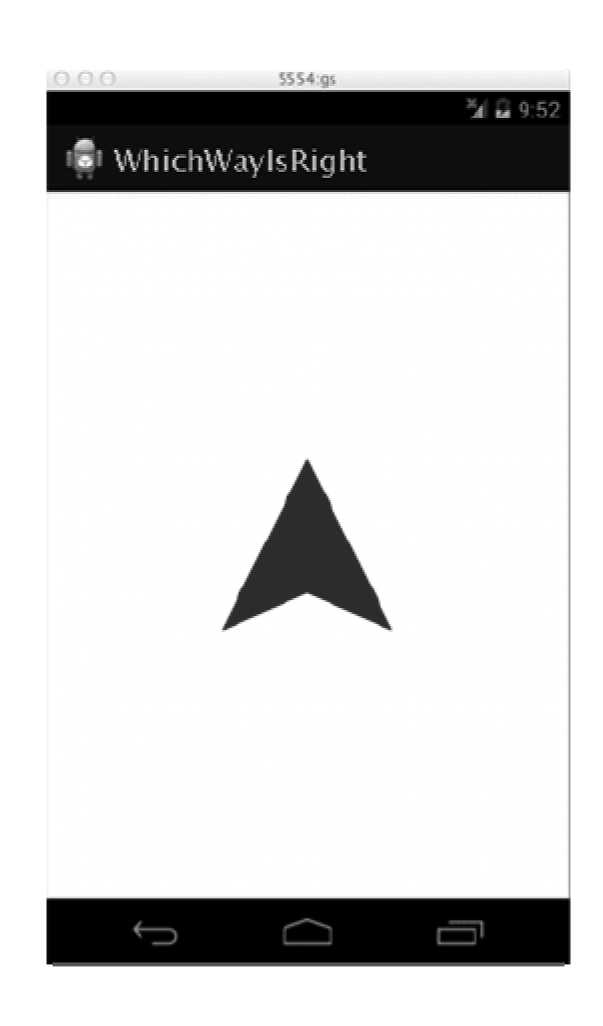


图 13-4　指南针程序效果

13.7　本章小结

本章介绍了 Android 平台所支持的传感器类型及传感器框架的基本使用。同时讲解了几种常见的传感器的编程要点。为了更深入了解传感器的编程特点，通过监测摇动和指南针两个案例对本章的重点知识进行了实践。